# 目　　录


**第 1 章　MATLAB 与通信仿真** ........ 1

1.1　MATLAB 简介 ........ 1

1.1.1　MATLAB 集成开发环境 ........ 2

1.1.2　MATLAB 编程语言 ........ 6

1.2　通信仿真 ........ 8

1.2.1　通信仿真的概念 ........ 8

1.2.2　通信仿真的一般步骤 ........ 9

**第 2 章　Simulink 入门** ........ 12

2.1　Simulink 简介 ........ 12

2.2　Simulink 工作环境 ........ 13

2.2.1　Simulink 模型库 ........ 13

2.2.2　设计仿真模型 ........ 14

2.2.3　运行仿真 ........ 14

2.2.4　建立子系统 ........ 15

2.2.5　封装子系统 ........ 17

2.3　Simulink 模型库 ........ 20

**第 3 章　S-函数** ........ 23

3.1　S-函数简介 ........ 23

3.1.1　S-函数的工作原理 ........ 23

3.1.2　S-函数基本概念 ........ 24

3.2　M 文件 S-函数 ........ 26

3.2.1　M 文件 S-函数简介 ........ 26

3.2.2　M 文件 S-函数的编写示例 ........ 30

3.3　C 语言 S-函数 ........ 46

3.3.1　C 语言 S-函数简介 ........ 46

3.3.2　C 语言 S-函数的编写示例 ........ 51

3.4　C++语言 S-函数 ........ 60

**第 4 章　信源和信宿** ........ 66

4.1　信源 ........ 66

4.1.1　压控振荡器 ........ 66

4.1.2　从文件中读取数据 ........ 68

4.1.3　数据源 ........ 72

4.1.4　噪声源 ........ 78

4.1.5　序列生成器 ........ 85

4.1.6 实例 4.1——通过压控振荡器实现 BFSK 调制 ........ 99
4.2 信宿 ........ 101
4.2.1 示波器 ........ 101
4.2.2 错误率统计 ........ 103
4.2.3 将结果输出到文件 ........ 105
4.2.4 眼图、发散图和轨迹图 ........ 108

**第 5 章 信道** ........ 116
5.1 加性高斯白噪声信道 ........ 116
5.1.1 函数 awgn() ........ 116
5.1.2 函数 wgn() ........ 118
5.1.3 加性高斯白噪声信道模块 ........ 120
5.1.4 实例 5.1——BFSK 在高斯白噪声信道中的传输性能 ........ 122
5.2 二进制对称信道 ........ 127
5.2.1 二进制对称信道模块 ........ 127
5.2.2 实例 5.2——卷积编码器在二进制对称信道中的性能 ........ 128
5.3 多径瑞利衰落信道 ........ 132
5.3.1 多径瑞利衰落信道模块 ........ 132
5.3.2 实例 5.3——BFSK 在多径瑞利衰落信道中的传输性能 ........ 134
5.4 伦琴衰落信道 ........ 138
5.4.1 伦琴衰落信道模块 ........ 138
5.4.2 实例 5.4——BFSK 在多径瑞利衰落信道中的传输性能 ........ 139
5.5 射频损耗 ........ 142
5.5.1 自由空间路径损耗模块 ........ 142
5.5.2 接收机热噪声模块 ........ 144
5.5.3 相位噪声模块 ........ 145
5.5.4 相位/频率偏移模块 ........ 146
5.5.5 I/Q 支路失衡模块 ........ 148
5.5.6 无记忆非线性模块 ........ 149

**第 6 章 信源编码** ........ 153
6.1 压缩和扩展 ........ 153
6.1.1 A 律压缩模块 ........ 153
6.1.2 A 律扩展模块 ........ 154
6.1.3 μ 律压缩模块 ........ 155
6.1.4 μ 律扩展模块 ........ 156
6.2 量化和编码 ........ 157
6.2.1 抽样量化编码器 ........ 157
6.2.2 触发式量化编码器 ........ 158
6.2.3 量化解码器 ........ 159
6.2.4 实例 6.1——A 律十三折与 μ 律十五折的量化误差 ........ 159

6.3 差分编码 ........ 162
6.3.1 差分编码器 ........ 162
6.3.2 差分解码器 ........ 163
6.4 DPCM 编码和解码 ........ 164
6.4.1 DPCM 编码器 ........ 164
6.4.2 DPCM 解码器 ........ 166
6.4.3 实例 6.2——DPCM 与 PCM 系统的量化噪声 ........ 166

**第 7 章 信道编码和交织 ........ 172**
7.1 分组编码 ........ 172
7.1.1 二进制线性码 ........ 172
7.1.2 二进制循环码 ........ 174
7.1.3 BCH 码 ........ 176
7.1.4 Reed-Solomon 码 ........ 178
7.1.5 Hamming 码 ........ 184
7.1.6 实例 7.1——Reed-Solomon 码在 CT2 中的应用 ........ 186
7.2 循环冗余码 ........ 192
7.2.1 CRC 编码器 ........ 192
7.2.2 CRC 检测器 ........ 195
7.2.3 实例 7.2——CRC-16 编码在 DECT 中的应用及其性能 ........ 197
7.3 卷积编码 ........ 202
7.3.1 卷积编码器 ........ 203
7.3.2 实例 7.3——IS-95 的卷积编码器 ........ 207
7.3.3 卷积译码器 ........ 211
7.3.4 实例 7.4——卷积码的软判决译码 ........ 214
7.4 块交织 ........ 220
7.4.1 通用块交织 ........ 220
7.4.2 矩阵交织 ........ 221
7.4.3 实例 7.5——交织器在 IS-95 中的应用 ........ 224
7.4.4 代数交织 ........ 228
7.4.5 随机交织 ........ 231
7.4.6 实例 7.6——cdma 2000 系统 Turbo 编码器的实现 ........ 232
7.5 卷积交织 ........ 249
7.5.1 复用交织 ........ 250
7.5.2 卷积交织 ........ 253
7.5.3 螺旋交织 ........ 255

**第 8 章 信号调制 ........ 259**
8.1 模拟幅度调制 ........ 259
8.1.1 双边带幅度调制 ........ 259
8.1.2 双边带抑制载波幅度调制 ........ 262

8.1.3 单边带幅度调制 ........ 265
8.2 模拟频率调制 ........ 268
8.2.1 基带频率调制 ........ 269
8.2.2 频带频率调制 ........ 270
8.3 模拟相位调制 ........ 272
8.3.1 基带相位调制 ........ 272
8.3.2 频带相位调制 ........ 274
8.4 数字幅度调制 ........ 275
8.4.1 基带脉幅调制 ........ 276
8.4.2 频带脉幅调制 ........ 278
8.4.3 基带正交幅度调制 ........ 281
8.4.4 频带正交幅度调制 ........ 282
8.4.5 基带矩形正交幅度调制 ........ 283
8.4.6 频带矩形正交幅度调制 ........ 285
8.4.7 实例 8.1——数字幅度调制的抗噪声性能 ........ 287
8.5 数字频率调制 ........ 291
8.5.1 基带 M 相频移键控调制 ........ 292
8.5.2 频带 M 相频移键控调制 ........ 293
8.6 数字相位调制 ........ 294
8.6.1 BPSK 调制 ........ 295
8.6.2 DBPSK 调制 ........ 296
8.6.3 QPSK 调制 ........ 297
8.6.4 实例 8.2——QPSK 在 IS-95 前向信道中的应用 ........ 299
8.6.5 DQPSK 调制 ........ 304
8.6.6 实例 8.3——DQPSK 在 USDC 中的应用 ........ 305
8.6.7 基带 OQPSK 调制 ........ 309
8.6.8 频带 OQPSK 调制 ........ 312
8.6.9 实例 8.4——OQPSK 在 IS-95 反向信道中的应用 ........ 314
8.6.10 基带 M-PSK 调制 ........ 318
8.6.11 频带 M-PSK 调制 ........ 319
8.6.12 基带 M-DPSK 调制 ........ 321
8.6.13 频带 M-DPSK 调制 ........ 323
8.7 数字连续相位调制 ........ 325
8.7.1 基带 CPM 调制 ........ 325
8.7.2 频带 CPM 调制 ........ 328
8.7.3 基带 MSK 调制 ........ 330
8.7.4 频带 MSK 调制 ........ 333
8.7.5 基带 GMSK 调制 ........ 335
8.7.6 频带 GMSK 调制 ........ 337
8.7.7 实例 8.5——GMSK 在 GSM 中的应用 ........ 338

8.7.8 基带 CPFSK 调制 ........ 342
8.7.9 频带 CPFSK 调制 ........ 343

**第 9 章 仿真和调试** ........ 346
9.1 运行仿真 ........ 346
9.1.1 设置仿真参数 ........ 346
9.1.2 运行仿真 ........ 356
9.2 调试和分析 ........ 358
9.2.1 调试仿真模型 ........ 358
9.2.2 分析仿真结果 ........ 364

**第 10 章 cdma 2000 移动通信系统** ........ 366
10.1 cdma 2000 系统简介 ........ 366
10.1.1 cdma 2000 1x 关键技术 ........ 367
10.1.2 cdma 2000 的信道划分 ........ 368
10.2 cdma 2000 反向业务信道 ........ 370
10.2.1 cdma 2000 反向业务信道简介 ........ 370
10.2.2 CRC 编码器 ........ 374
10.2.3 卷积编码器 ........ 379
10.2.4 信号交织器 ........ 384
10.2.5 正交扩频模块 ........ 392
10.2.6 PN 信号生成器 ........ 397
10.2.7 信号调制模块 ........ 404
10.2.8 初始化模块 ........ 406
10.2.9 基站接收器 ........ 410
10.3 cdma 2000 前向业务信道 ........ 412

# 第 1 章　MATLAB 与通信仿真

科学研究和工程应用中的数学计算（如矩阵运算、积分运算等）具有相当大的运算量，一般难以采用手工方式精确和快捷地完成。计算机和程序设计语言的出现使人们从繁琐的数值计算中解脱出来。若使用传统的程序设计语言（如 Basic、Fortran 和 C）编制计算程序，则要求程序设计人员对相关算法有深刻的了解，同时能够熟练地掌握编程语言的语法及编程技巧。对大多数科学工作者而言，同时具备这两方面技能是比较困难的，繁杂的程序设计过程不仅消耗人力与物力，而且影响工作进度和效率。为克服上述困难，1967 年美国 MathWorks 公司推出了 MATLAB（即 Matrix Laboratory 的简称）。MATLAB 自推出之后不断更新和扩充，目前已经推出 6.x 版本。

MATLAB 是一种功能强大的科学计算和工程仿真软件，它的交互式集成界面能够帮助用户快速地完成数值分析、矩阵运算、数字信号处理、仿真建模、系统控制和优化等功能。MATLAB 语言采用与数学表达相同的形式，不需要传统的程序设计语言，因而不像其他高级语言那样难于掌握。一般说来，用户可以在极短的时间内掌握 MATLAB 的基础知识，并且能够初步地应用 MATLAB 解决简单的问题。由于 MATLAB 的这些特性，它已经成为科研工作和工程仿真中的高效助手。

## 1.1　MATLAB 简介

MATLAB 软件系列产品是一套高效强大的工程技术数值运算和系统仿真软件，广泛应用于当今的航空航天、汽车制造、半导体制造、电子通信、医学研究、财经研究和高等教育等领域，被誉为“巨人肩膀上的工具”。研发人员借助 MATLAB 软件能迅速测试设计构想，综合评测系统性能，快速设计更好方案来确保更高技术要求。同时，MATLAB 也是国家教委重点提倡的一种计算工具。综合起来，MATLAB 有如下几个特点。

- 编程效率高

MATLAB 是一种面向科学与工程计算的高级语言，允许采用数学形式的语言编写程序，且比 Basic、Fortran 和 C 等语言更加接近我们书写计算公式的思维方式。用 MATLAB 编写程序犹如在演算纸上排列出公式与求解问题，因此，MATLAB 语言被称为“演算纸式”科学计算语言。

- 使用方便

MATLAB 语言是一种解释型语言，执行之前不需要进行专门的编译。一般情况下，在采用任何高级语言编写和调试程序时需要经历 4 个阶段，即编辑、编译、链接以及执行调试，并且这 4 个步骤之间是顺次执行的。MATLAB 语言与其他语言相比，较好地解决了上述问题，把编辑、编译、连接和执行融为一体。它能在同一画面上进行灵活操作快速排除输入程序中的书写错误、语法错误以至语意错误，从而加快了用户编写、修改和调试程序的速度，可以

说在编程和调试过程中它是一种比 VB 还要简单的语言。

具体地说，MATLAB 运行时，如直接在命令行输入 MATLAB 语句（命令），包括调用 M 文件的语句，每输入一条语句，就立即对其进行处理，完成编译、连接和运行的全过程。又如，将 MATLAB 源程序编辑为 M 文件，由于 MATLAB 磁盘文件也是 M 文件，所以编辑后的源文件就可直接运行，而不需进行编译和连接。在运行 M 文件时，如果有错，计算机屏幕上会给出详细的出错信息，用户经修改后再执行，直到正确为止。所以可以说，Matlab 语言不仅是一种语言，广义上讲是一种语言调试系统。

■ 扩充能力强

高版本的 MATLAB 语言有丰富的库函数，在进行复杂的数学运算时可以直接调用，而且 MATLAB 的库函数同用户文件在形成上一样，所以用户文件也可作为 MATLAB 的库函数来调用，用户可以根据自己的需要方便地建立和扩充新的库函数，以便提高 MATLAB 使用效率和扩充它的功能。另外，为了充分利用 Fortran、C 等语言的资源，包括用户已编好的 Fortran、C 语言程序，MATLAB 可以通过建立 MEX 文件的形式，混合编程，方便地调用有关的 Fortran 语言或 C 语言的子程序。

■ 语句简单，内涵丰富

MATLAB 语言中最基本最重要的成分是函数，其一般形式为 $[y_1, y_2, \ldots] = \mathrm{fun}(x_1, x_2, \ldots)$，即一个函数由函数名 fun，输入变量 $x_1$，$x_2$，…和输出变量 $y_1$，$y_2$，…组成。同一函数名 fun，不同数目的输入变量（包括无输入变量）及不同数目的输出变量代表着不同的含义（有点像面向对象中的多态性）。这不仅使 MATLAB 的库函数功能更丰富，同时大大减少了需要的磁盘空间，使得 MATLAB 编写的 M 文件简单、短小而高效。

■ 高效方便的矩阵和数组运算

MATLAB 语言像 Basic、Fortran 和 C 语言一样规定了矩阵的算术运算符、关系运算符、逻辑运算符、条件运算符及赋值运算符，而且这些运算符大部分可以毫无改变地照搬到数组间的运算。另外，它不需定义数组的维数，而且在 MATLAB 中，给出了矩阵函数、特殊矩阵专门的库函数，使之在求解诸如信号处理、建模、系统识别、控制、优化等领域的问题时，显得简捷、高效，这是其他高级语言所不能比拟的。在此基础上，高版本的 MATLAB 已逐步扩展到科学及工程计算的其他领域。

■ 方便的绘图功能

MATLAB 的绘图是十分方便的，它有一系列绘图函数（命令），例如线性坐标、对数坐标、半对数坐标和极坐标等，均只需调用不同的绘图函数（命令）。在图上标出图题、斜轴标注，格（栅）绘制也只需调用相应的命令，简单易行。另外，在调用绘图函数时调整自变量可绘出不变颜色的点、线、复线或多重线。这种为科学研究着想的设计是通用的编程语言所不及的。

## 1.1.1 MATLAB 集成开发环境

MATLAB 提供了一个集成化的开发环境，通过这个集成环境用户可以方便地设计仿真模型，执行仿真过程，并且分析仿真结果。图 1-1 所示是一个完整的 MATLAB 集成开发环境，其中包括 Command Window（命令窗口）、Workspace（工作区窗口）、Current Directory（当

前目录窗口）、Launch Pad（快速启动窗口）以及 Command History（历史命令窗口）。这些窗口既可以集成在 MATLAB 主窗口中，也可以单独成为一个窗口（通过单击窗口上方的按钮成为单独的窗口，或者通过各个独立窗口中的菜单命令 View / Dock 把窗口集成到 MATLAB 主窗口中）。

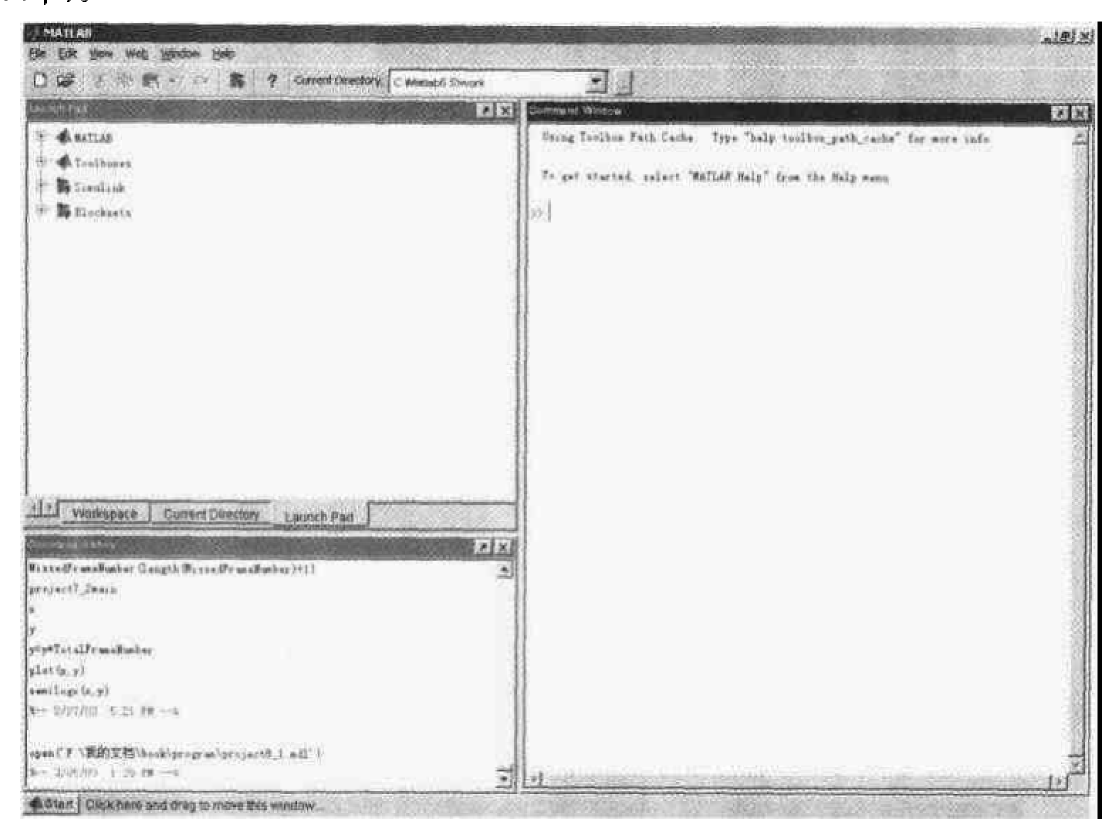



图 1-1　MATLAB 主窗口

■　Command Window（命令窗口）

命令窗口是 MATLAB 的主窗口，在出现命令提示符“>>”之后可以输入各种 MATLAB 命令，这些命令能够完成对 MATLAB 环境的设置，创建和设置仿真变量，运行仿真程序，启动 Simulink 仿真模块，以及对仿真数据的分析和处理等。

■　Workspace（工作区窗口）

MATLAB 工作区窗口以列表形式显示了 MATLAB 工作区中当前所有的变量名称及其属性，这些属性包括变量的长度、变量的类型及其占用的空间大小，如图 1-2 所示。

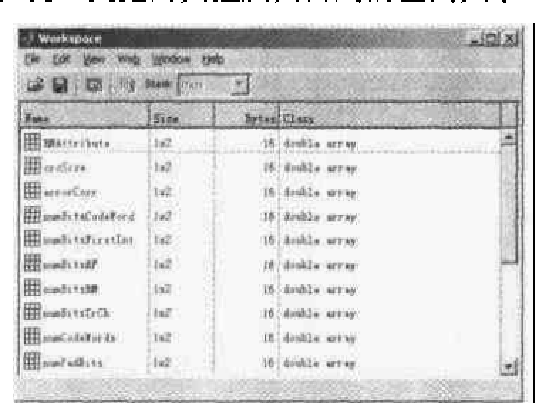



图 1-2　MATLAB 工作区窗口

在 MATLAB 工作区窗口中选择某个变量，单击按钮（或双击该变量）打开如图 1-3 所示的 Array Editor（数组编辑器）窗口。通过数组编辑窗口可以观察某个变量的数值。如果该变量是一个矩阵，则数组编辑器将显示这个矩阵中每个元素的数值。

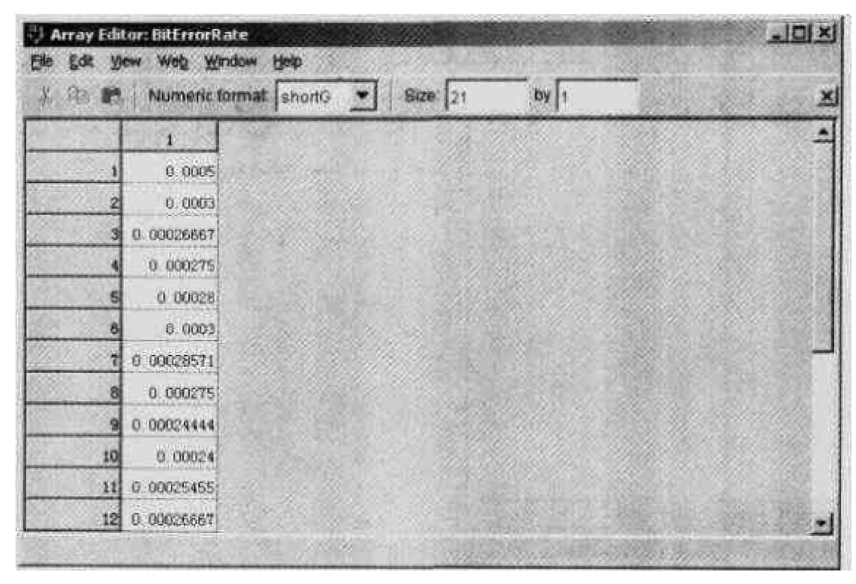



图 1-3　数组编辑器窗口

■　Current Directory（当前目录窗口）

当前目录窗口中显示了当前工作目录中的所有文件和文件夹，便于用户对当前目录中的文件进行管理。图 1-4 所示是当前目录窗口的示意图。

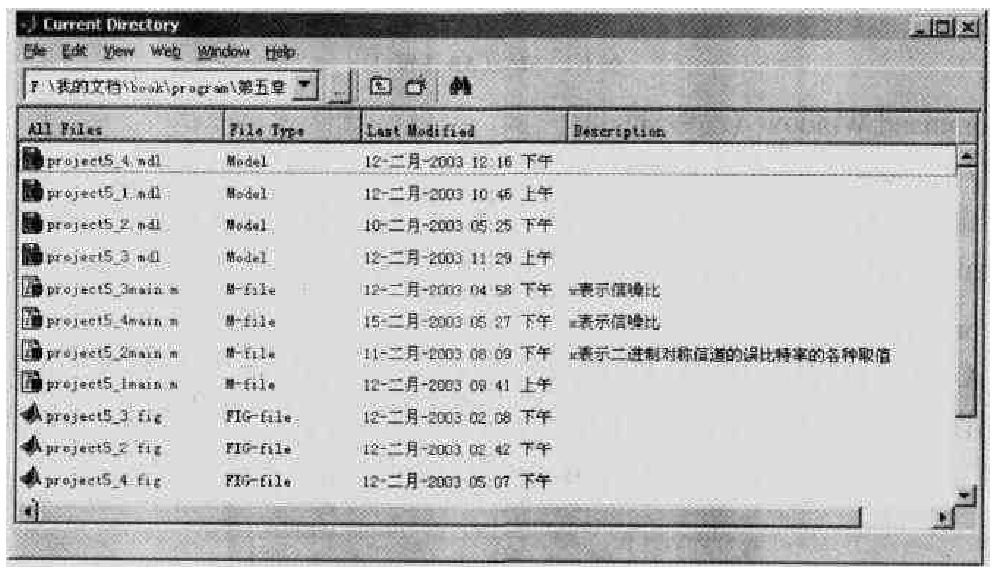



图 1-4　MATLAB 当前目录窗口

一般情况下，MATLAB 需要设置一个当前工作目录，使得 MATLAB 对所有文件的操作都默认在该目录中进行。如果 MATLAB 在当前目录中找不到指定的文件，它将根据环境变量中设定的路径进行搜索。如果搜索结束仍然找不到指定的文件，MATLAB 将发出错误警告。因此，在对一个没有在环境变量中标明的目录进行操作之前，应该把这个目录设置为当前目录。

■　Launch Pad（快速启动窗口）

快速启动窗口提供了一种打开 MATLAB 工具、文档和实例程序的便捷途径。在快速启动窗口中，各个条目组织成树型结构。单击某个目录前面的加号，MATLAB 将显示该目录下面的内容。图 1-5 所示是打开 Blocksets 目录以及 Blocksets 下面的 Communications 子目录后的快速启动窗口示例图。

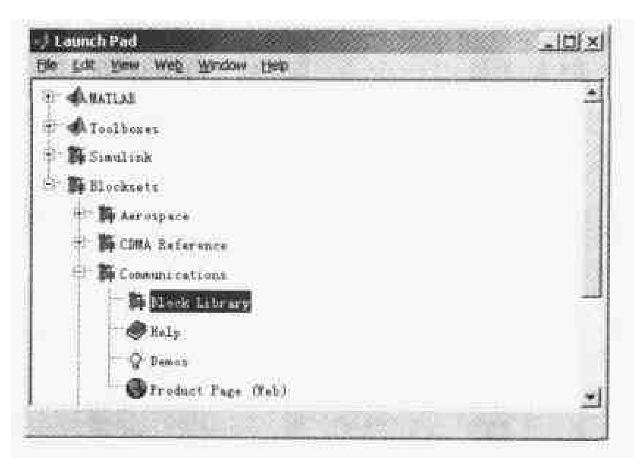



图 1-5　MATLAB 快速启动窗口

另外，单击 MATLAB 主窗口左下方的 Start 按钮也能够打开类似的快速启动菜单，如图 1-6 所示。这个快速启动菜单除了能够显示快速启动窗口的各个目录之外，还可以快速启动桌面工具管理、网页浏览、个人设置以及帮助文件等。

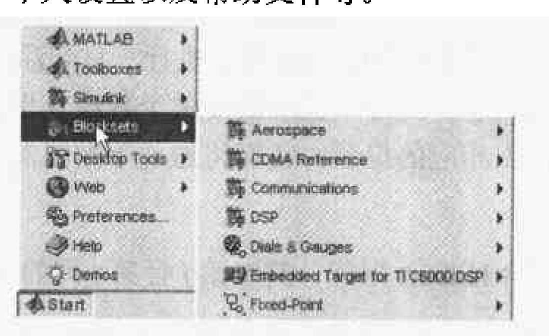



图 1-6　单击 Start 按钮之后地快速启动菜单

■　Command History（历史命令窗口）

历史命令窗口保存了用户在 MATLAB 命令窗口中输入的所有命令，并且记录了用户每次启动 MATLAB 的日期和时间，如图 1-7 所示。在历史命令窗口中选择一个或多个的命令（按住 Alt 或 Ctrl 按键可以选择多个命令）后双击鼠标左键，MATLAB 将自动把这些命令重新输入到命令窗口中运行。

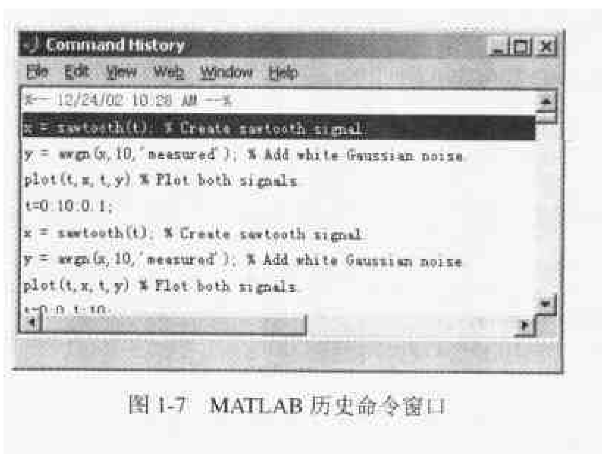



图 1-7　MATLAB 历史命令窗口

要清理命令窗口中的所有命令，选择菜单栏上的 Edit | Clear Command History，则 MATLAB 将删除所有的历史命令。

## 1.1.2 MATLAB 编程语言

MATLAB 程序大致分为两类，即 M 脚本文件（M-Script）和 M 函数（M-function），它们均是普通的文本文件。M 脚本文件中包含一组由 MATLAB 语言编写的语句，它类似于 DOS 下的批处理文件。M 脚本文件的执行方式很简单，用户只需在 MATLAB 的提示符“>>”下键入该 M 文件的文件名，MATLAB 就会自动执行该 M 文件中的各条语句，并将结果直接返回到 MATLAB 的工作区。M 函数格式是 MATLAB 程序设计的主流，一般情况下，不建议使用 M 函数格式编程。

MATLAB 的 M 函数是由 function 语句引导的，其基本格式如下：

```
function [返回变量列表] = 函数名 (输入变量列表)
[注释（由%引导）]
[检查输入变量和输出变量的格式]
[函数体语句]
```

在 M 函数中，输入变量和返回变量的个数分别由 nargin 和 nargout 两个变量确定，并且这两个变量是由 MATLAB 自动生成的，只要进入该函数就可以使用。如果输入变量的数目大于 1，则应该用方括号“[]”将它们包围起来，中间用逗号来分割。注释语句段的每行语句都应该由百分号“%”引导，百分号后面的内容不执行，只起注释作用。用户使用 help 命令可以显示出来注释语句段的内容。此外，正规的变量个数检查也是必要的。如果输入或返回变量格式不正确，则应该给出相应的提示。我们将通过下面的例子来演示函数编程的格式与方法。

假设要生成一个 $n \times m$ 阶 Hilbert 矩阵，其中第 $i$ 行第 $j$ 列的元素值等于 $1/(i+j-1)$。在这个 M 函数中，如果只有一个输入信号，则生成一个方阵（即 $m=n$）。同时，这个 M 函数具有参数检测功能，它在发现输入参数和输出参数的个数有错时给出错误信息。

```
function A = HilbertExample(n, m)
% HilbertExample ----   M-function Demonstration
%     A = HilbertExample (N, M) generates an N by M Hilbert matrix A.
%     A = HilbertExample (N) generates an N by N square Hilbert matrix.
%     HilbertExample (N,M) displays ONLY the Hilbert matrix, but do not return any
%          matrix back to the calling function.
%

% 输出信号的个数大于 1 时报错
if nargout>1
    error('Too many output arguments.');
end
% 只有一个输入信号时产生方阵
```

```
if nargin==1
    m = n;
% 否则，如果输入信号个数等于0或大于2时报错
elseif (nargin==0 | nargin>2)
    error('Wrong number of input arguments.');
end
% 产生一个n行m列的全零矩阵
B = zeros(n, m);
% 计算每个矩阵元素的数值
for i=1:n
    for j=1:m
        B(i,j)=1/(i+j-1);
    end
end
% 当输出信号个数等于1时返回这个矩阵
if nargout==1
    A = B;
% 否则，直接显示这个矩阵
elseif nargout==0
    disp(B);
end
```

保存这个M函数到文件HilbertExample.m中，然后把MATLAB的当前工作目录设置为这个M文件所在的目录（这点很重要，否则，MATLAB将提示找不到文件），这时候就可以运行这个M函数了。下面的程序段列出了针对这个M函数的各种操作及其结果。

```
>> help HilbertExample

  HilbertExample ----   M-function Demonstration
    A = HilbertExample (N, M) generates an N by M Hilbert matrix A.
    A = HilbertExample (N) generates an N by N square Hilbert matrix.
    HilbertExample (N,M) displays ONLY the Hilbert matrix, but do not return any
        matrix back to the calling function.

>> HilbertExample
??? Error using ==> hilbertexample
Wrong number of input arguments.
>> HilbertExample(3)
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
>> HilbertExample(3,4)
```

```
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
>> C=HilbertExample(3,4)
C =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
```

# 1.2 通信仿真

仿真是衡量系统性能的工具，它通过仿真模型的仿真结果来推断原系统的性能，从而为新系统的建立或原系统的改造提供可靠的参考。通过仿真，可以降低新系统失败的可能性，消除系统中潜在的瓶颈，防止对系统中某些功能部件造成过量的负载，优化系统的整体性能，因此，仿真是科学研究和工程建设中不可缺少的方法。

实际的通信系统是一个功能结构相当复杂的系统，对这个系统做出的任何改变（如改变某个参数的设置、改变系统的结构等）都可能影响到整个系统的性能和稳定。因此，在对原有的通信系统做出改进或建立一个新系统之前，通常需要对这个系统进行建模和仿真，通过仿真结果衡量方案的可行性，从中选择最合理的系统配置和参数设置，然后再应用于实际系统中。这个过程就是通信仿真。

## 1.2.1 通信仿真的概念

通信仿真是衡量通信系统性能的工具。通信仿真可以分成离散事件仿真和连续仿真。在离散事件仿真中，仿真系统只对离散事件做出响应，而在连续仿真中，仿真系统对输入信号产生连续的输出信号。离散事件仿真是对实际通信系统的一种简化，它的仿真建模比较简单，整个仿真过程需要花费的时间也比连续仿真少。虽然离散事件仿真舍弃了一些仿真细节，在有些场合显得不够具体，但仍然是通信仿真的主要形式。

与一般的仿真过程类似，在对通信系统实施仿真之前，首先需要研究通信系统的特性，通过归纳和抽象建立通信系统的仿真模型。图 1-8 所示是关于通信系统仿真流程的一个示意图。从图中可以看到，通信系统仿真是一个循环往复的过程，它从当前系统出发，通过分析建立起一个能够在一定程度上描述原通信系统的仿真模型，然后通过仿真实验得到相关的数据。通过对仿真数据的分析可以得到相应的结论，然后把这个结论应用到对当前通信系统的改造中。如果改造后通信系统的性能并不像仿真结果那样令人满意，还需要重新实施通信系统仿真，这时候改造后的通信系统就成了当前系统，并且开始新一轮的通信系统仿真过程。

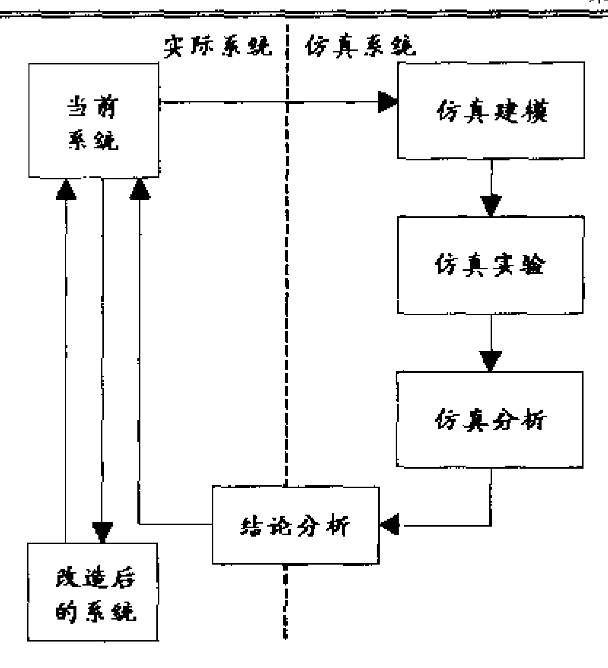



图1-8　通信系统仿真的流程

值得注意的是，在整个通信系统的仿真过程中，人为因素自始至终起着相当重要的作用。除了仿真程序的运行之外，通信仿真的每个步骤都需要进行人工干预，由人对当前的情况做出正确的判断。因此，通信仿真并不是一个机械的过程，它实际上是人的思维活动在计算机协助下的一种延伸。

## 1.2.2　通信仿真的一般步骤

通信系统仿真一般分成3个步骤，即仿真建模、仿真实验和仿真分析。应该注意的是，通信仿真是一个螺旋式发展的过程，因此这3个步骤可能需要循环执行多次之后才能够获得令人满意的仿真结果。

### 1. 仿真建模

仿真建模是根据实际通信系统建立仿真模型的过程，它是整个通信仿真过程中的一个关键步骤，因为仿真模型的好坏直接影响着仿真的结果以及仿真结果的真实性和可靠性。

仿真模型是对实际系统的一种模拟和抽象，但又不是完全的复制。简单的仿真模型容易被理解和操作，但是由于它忽略了很多关于实际系统的细节，因而在一定程度上影响了仿真的可靠性。如果仿真模型比较复杂，虽然它是对实际系统的一种忠实反映，但是其中包含了过多的相互作用因素，这些因素不仅需要消耗过多的仿真时间，而且使仿真结果的分析过程变得相当复杂。因此，仿真模型的建立需要综合考虑其可行性和简单性。在仿真建模过程中，我们可以先建立一个相对简单的仿真模型，然后再根据仿真结果和仿真过程的需要逐步增加仿真模型的复杂度。

仿真模型一般是一个数学模型。数学模型有多种分类方式，包括确定性（Deterministic）模型和随机（Stochastic）模型，静态（Static）模型和动态模型（Dynamic）。确定性模型的输入变量和输出变量都有固定数值，而在随机模型中，至少有一个输入变量是随机的。静态模型不需要考虑时间变化因素，动态模型的输入输出变量则需要考虑时间变化因素。一般情

况下通信仿真模型是一个随机动态系统。

在仿真建模过程中，首先需要分析实际系统存在的问题或设立系统改造的目标，并且把这些问题和目标转化成数学变量和公式。例如，我们可以设定改造后系统或新系统在达到系统最大容量时的误帧率，或者是通信系统的最大呼损率，等等。

有了这些具体的仿真目标之后，下一步是获取实际通信系统的各种运行参数，如通信系统占用的带宽及其频率分布，系统对于特定的输入信号产生的输出等。同时，对于通信系统中的各个随机变量，可以采集这些变量的数据，然后通过数学工具来确定随机变量的分布特性。

有了上面的准备工作，下一步就可以通过仿真软件来建造仿真模型了。最简单的工具是采用 C 语言等编程工具直接编写仿真程序，这种方法的优点是效率高，缺点则是不够灵活，没有一个易于实现的人机交互界面，不便于对仿真结果进行分析。除此之外，还可以采用专门的仿真软件建造仿真模型，比较常用的仿真软件包括 MATLAB、OPNET、NS2 等，这些软件具有各自不同的特点，适用于不同层次的通信仿真。例如，物理层仿真通常采用 MATLAB，而网络层仿真则适用采用 OPNET。

在完成仿真模型的软件实现之后，还需要对这个仿真模型的有效性进行初步的验证。一种简便的验证方法是采用特定的已知输入信号，这个输入信号分别通过仿真模型和实际系统，产生两种输出信号。如果仿真模型的输出信号与实际系统的输出信号比较吻合，说明这个仿真模型与原系统具有较好的相似性。当这两种输出信号差别很大时，最好先检查一下仿真模型的内部连接和设置，找出造成这种差异的原因。

仿真建模的最后一步是做好仿真模型的文档工作，这是最容易被大家忽略的。很多情况下，我们在完成系统的设计之后就迫不及待地运行仿真程序，待发现仿真结果与预期目标相差甚远时才回过头来焦头烂额地检查仿真模型的内部结构。这时候，往往原先的很多参数设置和条件假设都变得不可理解，这非常不利于修改参数和结构，不利于找错和排错。

### 2. 仿真实验

仿真实验是一个或一系列针对仿真模型的测试。在仿真实验过程中，通常需要多次改变仿真模型输入信号的数值，以观察和分析仿真模型对这些输入信号的反应，以及仿真系统在这个过程中表现出来的性能。需要强调的一点是，仿真过程中使用的输入数据必须具有一定的代表性，即能够从各个角度显著地改变仿真输出信号的数值。

实施仿真之前需要确定的另外一个因素是性能尺度。性能尺度指的是能够衡量仿真过程中系统性能的输出信号的数值（或根据输出信号计算得到的数值），因此，在实施仿真之前，首先需要确定仿真过程中应该收集哪些仿真数据，这些数据以什么样的格式存在，以及收集多少数据。

在明确了仿真系统对输入信号和输出信号的要求之后，最好把这些设置整理成一份简单的文档。编写文档是一个好习惯，它能够帮助我们回忆起仿真设计过程的一些细节。当然，文档的编写不一定要求很规范，并且文档大小应该视仿真设计的规模而定。

最后，还应该明确各个输入信号的初始设置以及仿真系统内部各个状态的初始值。仿真的运行实际上是计算机的计算过程，这个过程一般不需要人工干预，花费的时间由仿真的复杂度确定。如果需要比较仿真系统在不同参数设置下的性能，应该使仿真系统在取不同参数值时具有相同的输入信号（或相同的随机输入信号），这样才能够保证分析和比较的客观性和

可靠性。

对于需要较长时间的仿真，应该尽可能地使用批处理方式，使得仿真过程在完成一种参数配置的仿真之后能够自动启动针对下一个参数配置下一个仿真。这种方式减少了仿真过程中的人工干预，提高了系统利用率和仿真效率。

### 3. 仿真分析

仿真分析是一个通信仿真流程中的最后一个步骤。在仿真分析过程中，用户已经从仿真过程中获得了足够多的关于系统性能的信息，但是这些信息只是一些原始数据，一般还需要经过数值分析和处理才能够获得衡量系统性能的尺度，从而获得对仿真系统性能的一个总体评价。常用的系统性能尺度包括平均值、方差、标准差、最大值和最小值等，它们从不同的角度描绘了仿真系统的性能。

如果仿真过程需要一定的时间才能够达到平衡状态，在对输出数据进行分析和处理时一般要忽略最初的若干个数据，而只考虑平衡之后的输出。对于仿真尺度不随时间变化的平衡系统（Stationary System），还可能涉及到对输出变量稳定状态的求解。

另外一个需要注意的地方是，即使仿真过程中收集的数据正确无误，由此得到的仿真结果并不一定就是准确的。造成这种结果的原因可能是输入信号恰好与仿真系统的内部特性相吻合，或者输入的随机信号不具有足够的代表性。

图表是最简洁的说明工具，它具有很强的直观性，便于分析和比较，因此，仿真分析的结果一般都绘制成图表形式。我们使用的仿真工具一般都具有很强的绘图功能，能够便捷地绘制各种类型的图表。

以上就是通信仿真的一个循环。应该强调的是，仿真分析并不一定意味着通信仿真过程的完全结束。如果仿真分析得到的结果达不到预期的目标，用户还需要重新修改通信仿真模型，这时候仿真分析就成为了另外一个循环的开始。

本书在以后的章节里将结合 MATLAB 中的通信仿真模块库 Simulink 详细介绍通信仿真的各个过程。

# 第2章 Simulink入门

Simulink 是 MATLAB 中的一种可视化仿真工具，广泛应用于线性系统、数字控制、非线性系统以及数字信号处理的建模和仿真中。Simulink 采用模块化方式，每个模块都有自己的输入/输出端口，实现一定的功能。在 Simulink 中，仿真模型表现为若干个仿真模块的集合以及这些模块之间的连接关系，这就使得仿真的设计和分析过程变得相对直观和便捷，同时有利于仿真模型的扩充。

## 2.1 Simulink 简介

动态系统是输出信号随时间变化的系统。要描述这种系统的特性，传统的建模方法是先对系统的输入信号和输出信号进行分析，得到它们的系统方程，然后编写程序进行仿真。这种仿真方法有两个缺点。首先是不够直观，缺乏足够的人机交互。由于所有的输入信号和输出信号都被抽象成数值之间的关系，仿真表现为一种计算过程，因此难以对仿真的过程进行控制，也难以对仿真的输出数据进行直观的描述和分析。另外，这种方法缺乏系统性，尤其是在对复杂系统的处理过程中，难以采用模块化方法，从而降低了仿真程序的可读性和可扩展性。

Simulink 是 MATLAB 提供的用于对动态系统进行建模、仿真和分析的工具包。Simulink 提供了专门用于显示输出信号的模块，可以在仿真过程中随时观察仿真结果。同时，通过 Simulink 的存储模块，仿真数据可以方便地以各种形式保存到工作区或文件中，供用户在仿真结束之后对数据进行分析和处理。另外，Simulink 把具有特定功能的代码组织成模块的方式，并且这些模块可以组织成具有等级结构的子系统，因此具有内在的模块化设计要求。基于上述优点，Simulink 称为一种通用的仿真建模工具，广泛应用于通信仿真、数字信号处理、模糊逻辑、神经网络、机械控制和虚拟现实等领域。

根据输出信号与输入信号的关系，Simulink 提供 3 种类型的模块：连续模块、离散模块和混合模块。连续模块是指输出信号随着输入信号发生连续变化的模块，离散模块则是输出信号以固定间隔变化的模块。对于连续模块，Simulink 采用积分方式计算输出信号的数值，因此，连续模块主要涉及导数的计算及其积分。离散模块的输出信号在下一个抽样时刻到来之前保持恒定，这时候 Simulink 只需以一定的间隔计算输出信号的数值。混合模块是根据输入信号的类型来确定输出信号类型的，它既能够产生连续输出信号，也能够产生离散输出信号。

如果一个仿真模型中只包含离散模块，这时候 Simulink 采用固定步长方式进行仿真（即每个一定的间隔计算一次输出信号）。当所有的离散模块都有相同的抽样间隔时，Simulink 只需要按照这个间隔实施仿真；否则，Simulink 采用多速率方式进行仿真。多速率仿真模式的一种方案是选取一个最大可用间隔，使之适用于所有的离散模块。这个间隔一般是各个离散模块抽样间隔的最大公约数。对于可变步长方式，多速率仿真模式按照各个模块的抽样间

隔列出系统可能的仿真时刻，在仿真时刻到来的时候只对相应的离散模块实施仿真，从而在一定程度上提高了仿真的效率。

如果仿真模型中包含了连续模块，Simulink 将采用连续方式对模块进行仿真。如果模块中既包含了连续模块，又包含了离散模块，Simulink 采用两种仿真步长进行仿真。对于其中的离散模块，Simulink 可以按照离散模块的方式进行仿真，这个仿真步长称为主步长（major step size）。在每个主步长仿真中，Simulink 使用小步长间隔（minor step size），通过积分运算得到连续状态的当前输出信号。

## 2.2　Simulink 工作环境

当采用 Simulink 进行建模和仿真时，一般是从 Simulink 模型库中提供的模块出发，通过组合各种模块来完成模块的设计。Simulink 模型库提供了一种模块的集成环境，通过它可以快速地开发各种仿真模型。

### 2.2.1　Simulink 模型库

在 MATLAB 工作区中输入“simulink”并回车，或者单击 MATLAB 工具栏上的按钮，就进入了如图 2-1 所示的 Simulink 模型库。

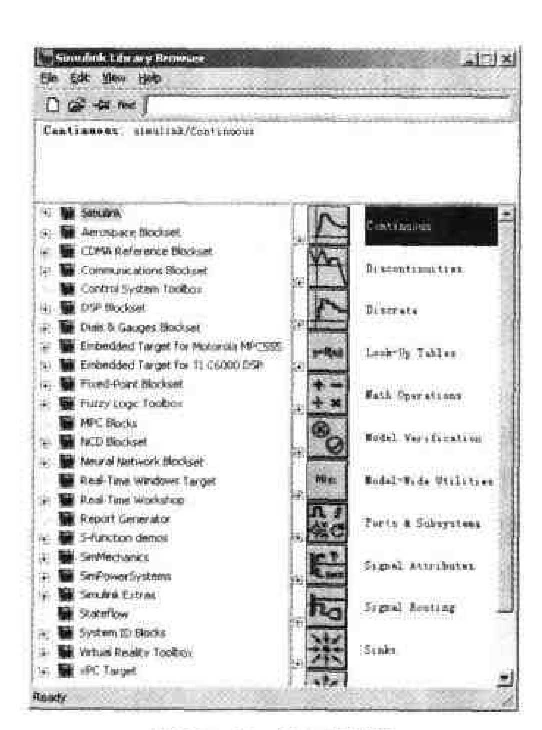



图 2-1　Simulink 模型库

Simulink 模型库中的仿真模块组织成一个三级树型结构，例如，图 2-1 所示 Simulink 子模型库包含了 Continuous、Discontinuous、Discrete 等下一级的模型库，其中 Continuous 模型库中包含了若干个模块，这些模块可以直接加入到自己的仿真模型中。

## 2.2.2 设计仿真模型

在 MATLAB 主窗口或 Simulink 模型库的菜单栏中依次选择“File” | “New” |“ Model”，MATLAB 生成一个空白的仿真模型窗口，如图 2-2 所示。

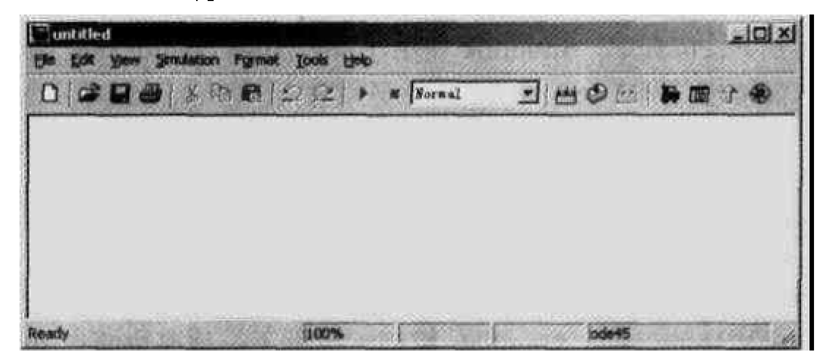



图 2-2 MATLAB 产生的空白仿真模型

在设计仿真模型的过程中，如果 Simulink 模型库中包含了仿真模型所需的模块，则在 Simulink 模型库中选中这个模块，单击鼠标右键，从浮动菜单中选择 “Add to untitled”（或直接把模块拖到仿真模型中），这时候就把这个选中的模块加入到仿真模型中了。

Simulink 模型库窗口提供了模块查找功能。在 Simulink 模型库窗口的工具栏上单击按钮，弹出如图 2-3 所示的模块查找对话框。输入所需查找的模块名称的关键字，单击 “Find Next” 按钮，则 Simulink 自动搜索整个模型库。

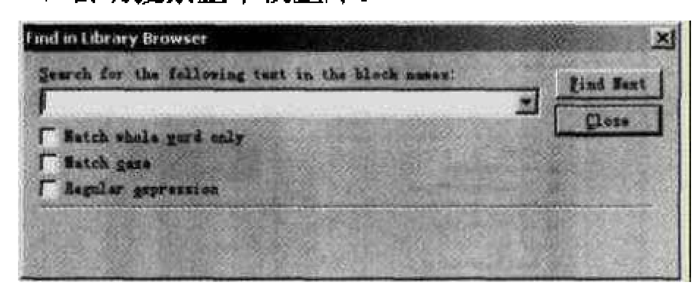



图 2-3 Simulink 模型库的模块查找对话框

Simulink 模型库中的模块一般具有各种参数设置。在仿真模型窗口中双击模块，弹出该模块的参数设置对话框，这时候可以修改模块中各个参数的数值。通常情况下，仿真模块的设计过程就是对 Simulink 模型库中各个模块的一种组合。如果 Simulink 模型库中没有所需的模块，这时候可以通过 S-函数构造自己的模块，并且把这个模块与其他 Simulink 模块组合起来，实现相应的仿真功能。关于 S-函数的内容，将在第 3 章进行详细论述。

## 2.2.3 运行仿真

仿真模型有两种运行方式：菜单方式和命令行方式。在 Simulink 中打开仿真模型，然后在菜单栏中依次选择 “Simulation” | “Start”，或者在工具栏上单击按钮，则仿真模型将以菜单方式运行。菜单方式的优点在于它的交互性，通过在仿真模型中设置示波器模块（Scope）或显示模块（Display）可以在仿真过程中观察输出信号的数值。同时，有些仿真

模块还允许用户在不中断仿真进程的条件下随时更改模块的参数设置。图 2-4 所示是一个正在运行的仿真模型，其中状态栏显示了仿真的进度和时间。

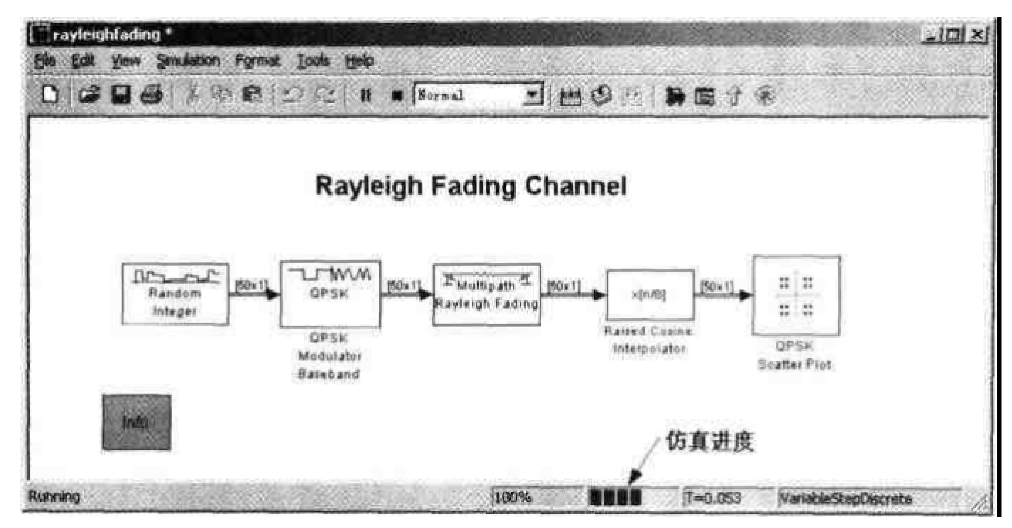



图 2-4 一个正在运行的仿真模型

命令行方式一般用于执行批处理方式的仿真，它是通过 MATLAB 命令“sim”启动仿真进程，例如，在 MATLAB 工作区中输入“sim rayleighfading”命令后，Simulink 开始运行仿真模型 rayleighfading。如果需要多次运行仿真程序，可以把这些命令编写成一个 M 文件，然后在 MATLAB 工作区中执行这个 M 文件就可以了。这种方法广泛地应用于本书后面各个章节的实例程序中。用命令行方式启动仿真模型后，Simulink 并不自动打开相应的模型，因此不能直接观察仿真的进程，但是仍然可以通过各种显示模块观察输出信号。

MATLAB 把工作区和 Simulink 集成在一起，因此仿真模型的两种运行方式可以交互使用。一般情况下，仿真结果保存到工作区中，用户可以在仿真结束之后对仿真结果进行分析和加工，并且根据仿真数据绘制各种图表。

## 2.2.4 建立子系统

在 Simulink 中，若干个仿真模块可以组合起来构成一个子系统（Subsystem），以减少每个仿真模型窗口中显示的仿真模块的数目。采用子系统的另外一个好处是可以把复杂的仿真模型按照功能关系组织成等级结构，使得每个子系统都是一个相对独立的功能实体，这些子系统组合起来构成一个功能更强的子系统。

Simulink 子系统可以按照自顶向下的方式进行构造。首先，在仿真模型中添加一个子系统模块（Subsystem），设置这个子系统模块的名称，然后双击该模块，待打开该模块之后就可以对子系统的功能进行设计了。子系统的输入、输出分别由 Simulink 模型库中的输入端口模块（In）和输出端口模块（Out）构造，在设计子系统的过程中可以根据需要添加若干个这样的输入输出模块。

另外一个构造子系统的方法是自底向上，即先在同一个窗口中添加各个 Simulink 模块，当模块的数目增加到一定的限度时，可以选中其中若干个模块（按住 Shift 键后单击各个需要组合的模块），然后单击鼠标右键，在弹出式菜单中选择 Create Subsystem，这时候就创建了一个子系统，并且 Simulink 自动为这个子系统设置了输入端口和输出端口。在 Simulink 中，子系统和输入、输出端口的名称都可以由用户自行设置。

子系统的命名方式类似于系统目录的命名，每个子系统的名称是它所在的上一级系统名称与子系统名称的组合，中间以反斜杠（"/"）分隔。例如，"project5_1/Source"表示 project5_1 模型中名为 Source 的子系统。

用户可以设置子系统在双击之后的打开方式。选择菜单栏上的"File"|"Preferences..."命令打开如图 2-5 所示的对话框，在左边的列表框中选择"Simulink"，这时候可以在右边面板的"Window reuse"下拉列表中选择所需的方式。

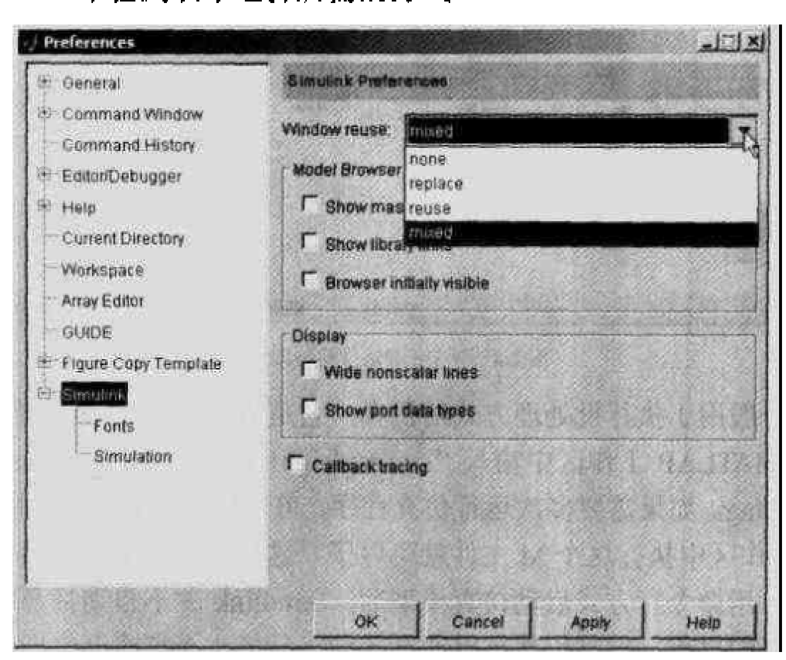



图 2-5 MATLAB 的 Preferences 对话框

如果把"Window reuse"设置为"none"，双击子系统时，Simulink 在新窗口中打开子系统，原窗口保持不变；如果"Window reuse"设置为"reuse"，Simulink 在当前窗口中打开子系统；如果"Window reuse"设置为"replace"，Simulink 在新窗口中打开子系统，同时原窗口消失；如果"Window reuse"设置为"mixed"，Simulink 在新窗口中打开子系统，同时保持原窗口不变，这种方式与 none 的差别在于，当在子系统中选择菜单栏上的"View"|"Go to parent"命令返回上一级系统时，none 方式中这个子系统窗口不会消失，而 mixed 方式的子系统窗口被自动隐藏。Simulink 还提供了子系统模块的另外一种显示方式：模型浏览器方式，如图 2-6 所示。

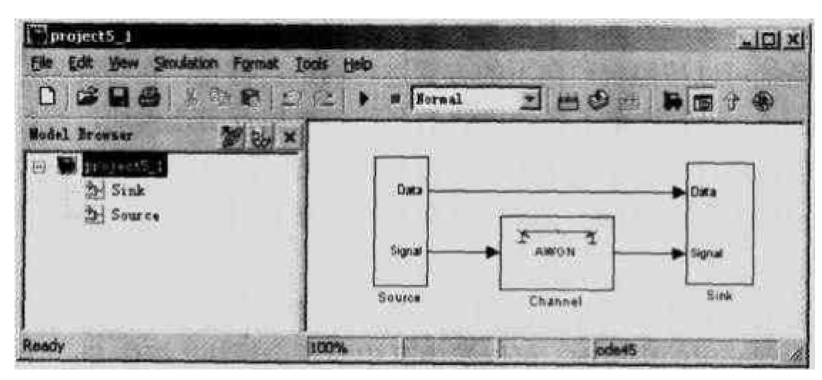



图 2-6 仿真模型浏览器

在这种方式中，窗口左边一列以树型结构显示出本模型中的所有子系统及其下一级的子系统。选中其中的某个子系统之后，右边的窗口自动显示出这个子系统的内部结构。这种显

示方式与 Windows 浏览器比较类似，其优点是便于在各个子系统中实现快速切换。要切换仿真模型的浏览器方式，依次选择菜单栏上的“View”|“Model browser options”|“Model browser”命令，或者单击工具栏上的按钮。

## 2.2.5 封装子系统

Simulink 模块库中的每一个模块都有它独特的图标和参数设置对话框。对于用户自己创建的子系统，也可以通过 Simulink 来设置该模块的图标和参数。选中某个子系统，单击右键后从弹出式菜单中选择“Mask subsystem”（或在菜单栏上选择“Edit”|“Mask subsystem...”），这时候就创建了一个封装子系统（Masked Subsystem）。需要注意的是，对于一个已经被封装的子系统，它的 Mask subsystem 命令无效。

子系统转化成封装子系统之后，Simulink 弹出一个封装编辑器对话框，这时候可以设置封装子系统的图标和模块参数。封装编辑器共有 4 个属性页，分别用与设置模块的图标、参数、初始化代码以及文档信息。图 2-7 所示是其中的 Icon 属性页。在这个属性页中，读者可以设置模块的外观属性，并且可以在 Drawing commands 编辑框中输入代码，以绘制模块的图标。在图 2-7 所示的窗口中，我们通过 MATLAB 命令行“disp（'Source'）”在模块中显示一行文字“Source”。

图 2-8 所示是封装编辑器的 Parameters 属性页。通过 Parameters 属性页读者可以设置封装子系统的自定义参数，这些参数将出现在封装子系统的参数设置对话框中。对于每一个自定义参数，它有一个用于在参数设置对话框中显示的名称（Prompt）和一个变量名称。每个自定义参数可以通过编辑框（Edit）、复选框（Checkbox）或者是下拉列表（Popup）设置参数值。如果选择了 Evaluate 复选框，则该参数的输入将自动转化成数值传递给相应的变量；否则，Simulink 自动把字符串传递给这个变量。如果选择了 Tunable 复选框，则该参数的数值可以在仿真过程中动态改变。例如，对于名为 Samples per Symbol 的自定义参数，它的内部变量名为 SamplesPerSymbol，用户可以通过下拉列表“popup”来选择该参数的数值 2、3 或 4，同时 Simulink 传递给变量 SamplesPerSymbol 的是一个数值，并且该参数可以在仿真过程中动态改变。

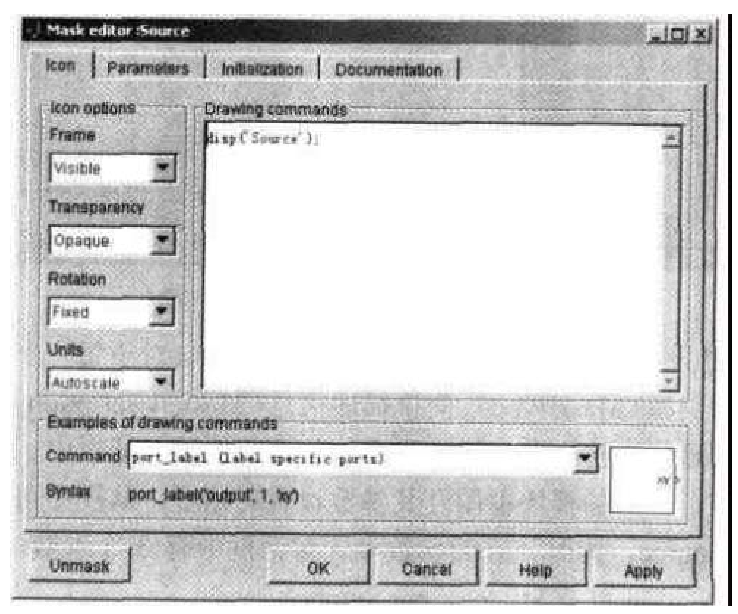



图 2-7 封装编辑器的 Icon 属性页

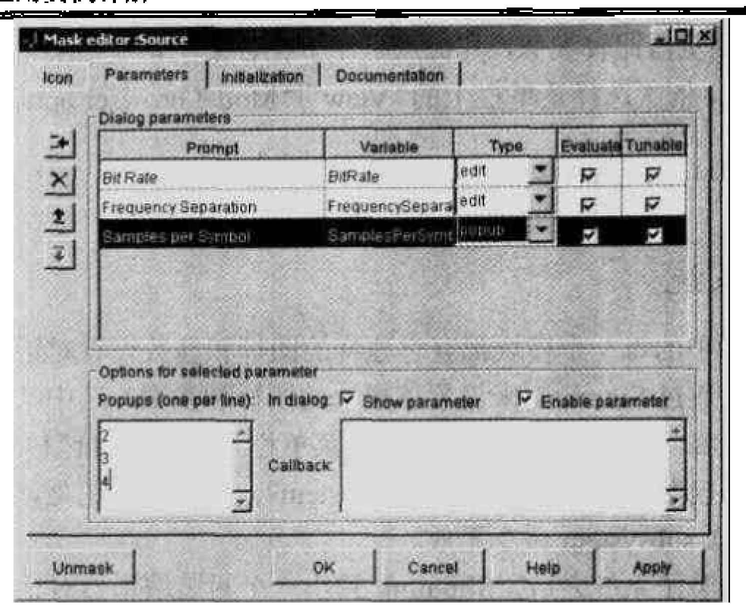



图 2-8 封装编辑器的 Parameters 属性页

图 2-9 所示是封装编辑器的 Initialization 属性页，在这里，读者可以对每个内部变量实施初始化（或执行其他初始化命令）。封装子系统的内部变量可以作为这个子系统中的子模块的参数值，因此，通过对子系统进行封装可以方便地实现对子系统中各个模块参数的控制。在图 2-9 所示中，分别把变量 BitRate、FrequencySeparation 以及 SamplesPerSymbol 的初始值设置为 10000、24000 和 2，这些参数能够用作内部各个子模块的参数值。

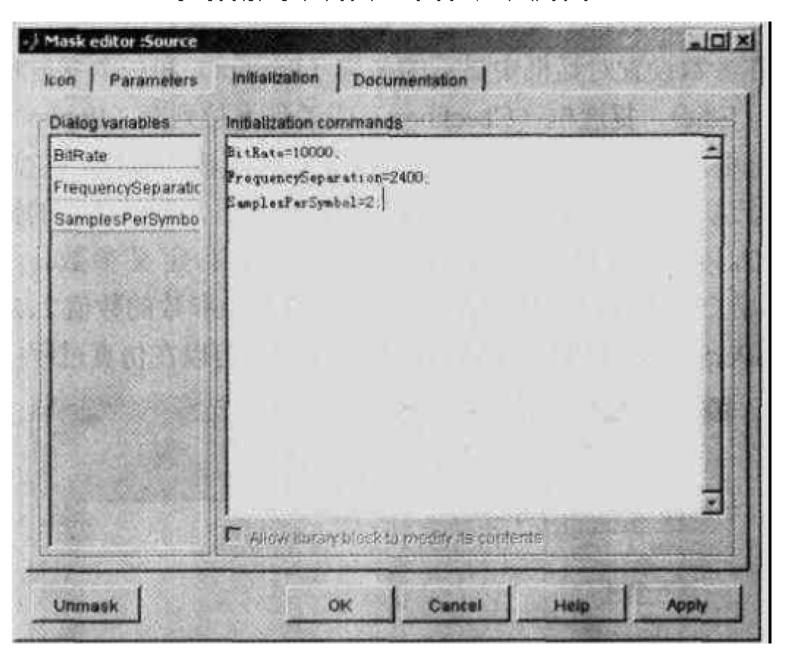



图 2-9 封装编辑器的 Initialization 属性页

图 2-10 所示则是封装编辑器的 Documentation 属性页。对于每一个封装子系统，读者可以设置这个子系统的类型（Mask type）、简单描述信息（Mask description）以及帮助信息（Mask help），其中类型和描述信息将出现在封装子系统的参数设置对话框中。图 2-11 所示是完成上述参数设置之后封装子系统的模块框图及其参数设置对话框。从图中可以看到，这个封装子系统的模块名称为 Source，同时，它的参数设置对话框中有 3 个参数，参数名称分别为 Bit Rate、Frequency Separation 以及 Samples per Symbol。

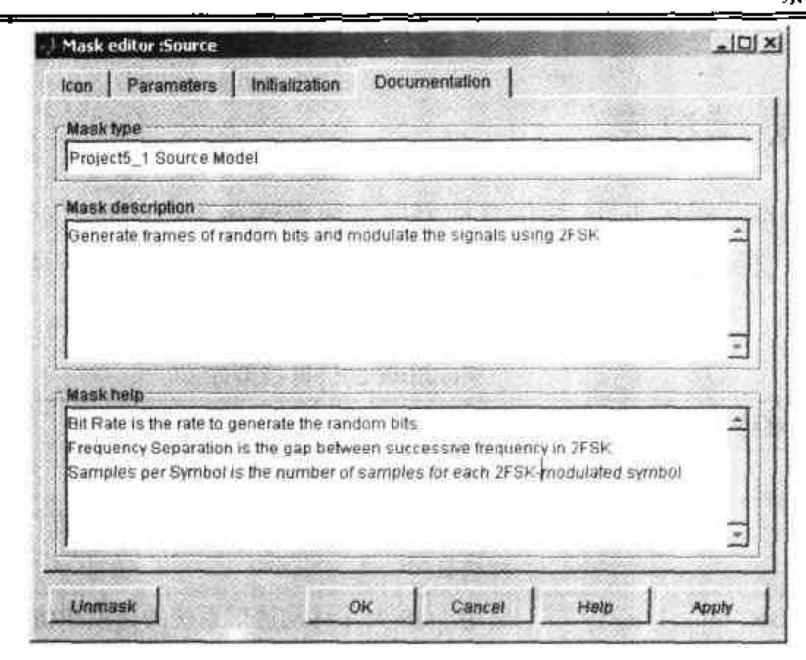



图 2-10　封装编辑器的 Documentation 属性页

在图 2-11 所示的参数设置对话框中单击 Help 按钮，打开如图 2-12 所示的帮助窗口，窗口中显示的内容就是我们在封装编辑器 Documentation 的属性页中填写的帮助信息（Mask help）。

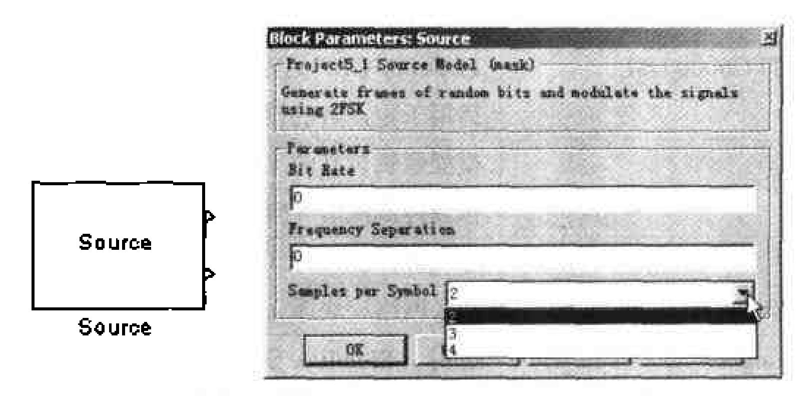



图 2-11　完成参数设置之后的子系统及其参数设置对话框

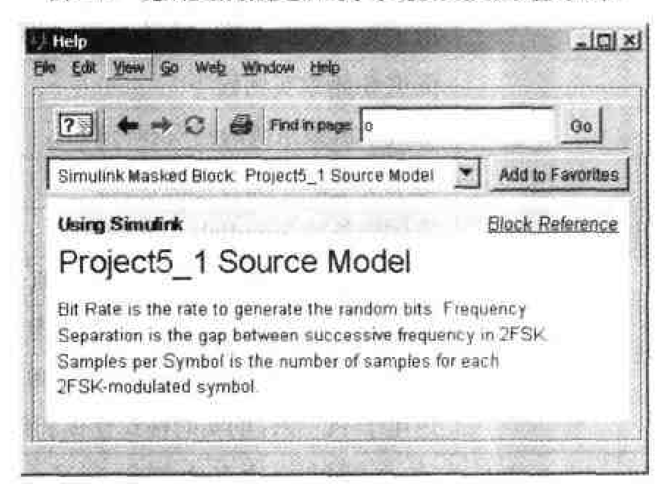



图 2-12　封装子系统的帮助信息

封装子系统在很大程度上方便了子系统中各个模块的参数设置过程，提供了一种更完善的系统集成模式，这种特性在构建复杂子系统的过程中显得尤为重要。

## 2.3 Simulink 模型库

Simulink 5.0 版本共提供了 25 种仿真模型库，内容涵盖了通信仿真、数字信号处理、模糊逻辑、神经网络、机械控制和虚拟现实等。表 2-1 列出了 Simulink 模型库中的各个子模型库及其介绍。

**表 2-1　　Simulink 的仿真模型库**

| 模型库名称 | 说明 |
| --- | --- |
| Simulink | 提供基本的 Simulink 建模和仿真模块 |
| Aerospace Blockset | 对飞机、导弹、武器和航空动力系统进行建模、分析、集成和仿真 |
| CDMA Reference Blockset | 适用于 IS-95A 移动通信系统的建模和仿真 |
| Communications Blockset | 适用于通信系统的建模和仿真 |
| Control System Toolbox | 适用于反馈控制系统的设计和仿真 |
| DSP Blockset | 适用于数字信号处理（DSP）的设计和仿真 |
| Dials & Gauges Blockset | 提供以图形方式控制信号和仿真参数的模块 |
| Embedded Target for Motorola MPC555 | Motorola MPC555 的内置代码 |
| Embedded Target for TI C6000 DSP | TI C6000 DSP 的内置代码 |
| Fixed-Point Blockset | 适用于定点数的设计和仿真 |
| Fuzzy Logic Toolbox | 模糊逻辑仿真的工具箱 |
| MPC Blocks | 预测控制模型（MPC，Model Predictive Control）工具箱 |
| NCD Blockset | 非线性控制设计（NCD，Nonlinear Control Design）模块库 |
| Neural Network Blockset | 神经网络模块库 |
| Real-Time Windows Target | 适用于 Simulink 模型和 Stateflow 模型在 PC 上的实时仿真 |
| Real-Time Workshop | 根据 Simulink 模型产生 C 语言代码 |
| Report Generator | 用于为 Simulink 模型和 Stateflow 模型自动生成文档 |
| S-function demos | 提供 S-函数实例程序 |
| SimMechanics | 适用于机械系统的建模和仿真 |
| SimPowerSystems | 适用于电源系统的建模和仿真 |
| Simulink Extras | 提供额外的 Simulink 仿真模块 |
| Stateflow | 适用于事件驱动系统的建模和仿真 |
| System ID Blocks | 根据输入和输出信号建立线性动态系统 |
| Virtual Reality Toolbox | 适用于对虚拟现实系统进行建模和仿真 |
| xPC Target | 适用于 PC 硬件的实时快速建模和仿真 |

从表 2-1 中可以看到，Simulink 在各个领域的仿真中都得到了广泛的应用。本书主要介绍 Simulink 模型库中的通信仿真模块库（Communications Blockset），着重介绍通信仿真模块在移动通信系统的建模和仿真中的应用。

图 2-13 所示是 Simulink 中的通信仿真模块库，它由 13 个子模块库组成。

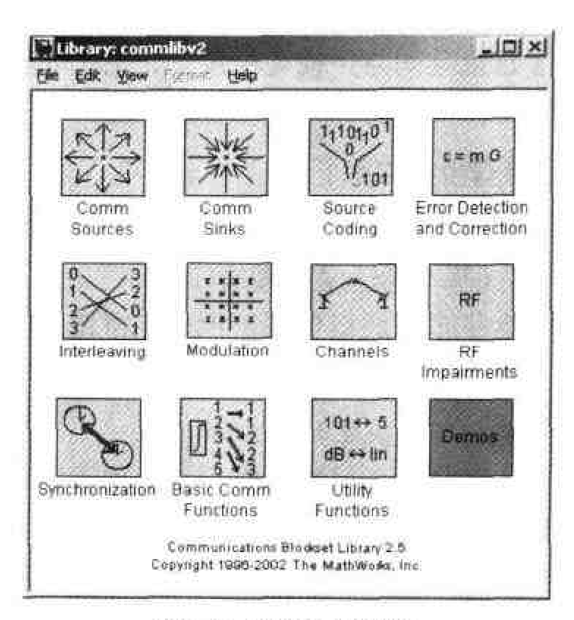



图 2-13 通信仿真模块库

■ Comm Sources（信源模块库）

信源模块库提供了多种信源模块，这些模块可以分成 4 类，即受控信源模块（Controlled Sources）、数据源模块（Data Sources）、噪声产生器模块（Noise Generators）以及序列生成器模块（Sequence Generators）。

■ Comm Sinks（信宿模块库）

信宿模块库共有 6 种信宿模块，主要用于绘制信号的眼图、发散图和轨迹图，计算信号的误帧率或误比特率，以及把数据保存到文件中。对于其他常用的信宿模块，可以在 Simulink 模块库的信宿子模块库中找到。

■ Source Coding（信源编码模块库）

信源编码模块库中包含各种用于实现抽样和量化功能的模块，包括 A 律压缩和扩展模块、μ 律压缩和扩展模块、量化编码器模块以及差分脉冲编码调制模块等。

■ Error Detection and Correction（信道编码模块库）

信道编码模块库中包含各种用于实现信道编码的模块，它分成 3 个子模块库，即分组编码模块库、卷积编码模块库以及循环冗余码模块库。

■ Interleaving（信号交织模块库）

信号交织模块库中包含各种用于实现信号交织功能的模块，其中包括多种块交织模块和卷积交织模块库。

■ Modulation（信号调制模块库）

信号调制模块库中包含了各种用于实现信号调制和解调功能的模块，这些模块可以实现模拟基带信号调制、模拟频带信号调制、数字基带信号调制和数字频带信号调制。

■ Channels（信道模块库）

信道模块库中包含了 4 种常见的信道模块，即加性高斯白噪声信道模块、二进制对称信道模块、多径瑞利衰落信道模块以及伦琴衰落信道模块。

■ RF Impairments（射频损耗模块库）

射频损耗模块库中包含了自由空间路径损耗模块、无记忆非线性模块、相位噪声模块、接收机热噪声模块等，用于对射频信号的各种损耗进行仿真。

■ Synchronization（信号同步模块库）

信号同步模块中包含了 4 种锁相环模块，用于对信号同步功能进行仿真。

■ Basic Comm Functions（基本通信函数模块库）

基本通信函数模块库中包含了多种通信仿真过程中经常使用的模块，这些模块分成两大类，即积分器模块和序列操作模块。

■ Utility Functions（公用函数模块库）

公用函数模块库中包含了 6 种常用的转换函数模块，用于实现信号的单极性与双极性之间的转换、十进制数与二进制数之间的转换等。

本书后面的各个章节将结合移动通信系统中的实例对这些模块库中的各个模块进行详细的介绍。

# 第3章　S-函数

S-函数是系统函数（System-functions）的简称。在很多情况下，Simulink 模型库（Simulink Library）中的模块不能完全满足用户的要求，这时候需要由用户自己来编写相应的代码。M 文件虽然能够用来编写 MATLAB 函数代码，但是它不具备与 Simulink 的接口，因此难以与 Simulink 其他模块一起使用。S-函数则提供了函数代码与 Simulink 之间的接口，使得用户编写的代码既能够像 Simulink 模型库中的模块那样具有统一的仿真接口，同时能够实现各种灵活的控制和计算功能。从这个意义上说，S-函数是对 Simulink 模块库功能的扩展。S-函数的代码既可以用 MATLAB 语言编写，也可以用其他通用的编程语言（如 C、C++、Ada 或 Fortran 等）编写，后者具有更强的控制能力，它们被编译成 MEX（MATLAB EXecutable）文件，并且在仿真过程中动态装载。

## 3.1　S-函数简介

通过 S-函数可以方便地编写仿真代码以创建自己的 Simulink 模块，因此，S-函数是对 Simulink 模块库功能的扩展。根据 S-函数代码使用的编程语言，S-函数可以分成 M 文件 S-函数（即用 MATLAB 语言编写的 S-函数）、C 语言 S-函数、C++语言 S-函数、Ada 语言 S-函数以及 Fortran 语言 S-函数等。通过 S-函数创建的模块具有与 Simulink 模型库中的模块相同的特征，它可以与 Simulink 求解器进行交互，支持连续状态和离散状态模型。

### 3.1.1　S-函数的工作原理

每个 Simulink 模块都可以表示成输入信号 $u$ 、输入信号 $y$ 以及内部状态 $x$ 之间的关系，如图 3-1 所示。

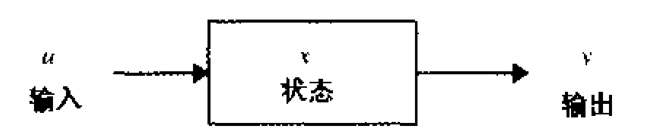



图 3-1　Simulink 模块的基本结构

在某个时刻 $t$，Simulink 模块的内部状态 $x$ 由两部分组成：连续状态 $x_c$ 和离散状态 $x_d$，且 $x = x_c + x_d$，此时输出信号 $y = f_0(t,x,u)$，连续状态的导数 $x_c' = f_d(t,x,u)$，离散状态 $x_{d_{k+1}} = f_u(t,x,u)$。Simulink 根据连续状态导数方程进行积分运算，得到各个连续状态的数值，同时通过离散状态方程计算离散状态的当前值。这样，Simulink 就可以得到各个时刻的状态及其输出信号，实现对仿真结果的求解。

在仿真过程中，每个 Simulink 模块的执行过程可以分成 3 个阶段：初始化阶段、仿真循

环阶段和仿真结束阶段。在初始化阶段，Simulink 把各个模块调入内存，检查模块的数据类型和长度，设置仿真时间间隔，制订仿真模块的执行顺序，以及内存分配。在仿真循环阶段，Simulink 按照初始化阶段制定的顺序依次执行各个模块，计算当前时刻的离散状态和输出信号，以小步长积分的方式计算各个连续状态的数值以及由此产生的输出。这个过程一直持续到仿真过程结束，然后 Simulink 进入仿真结束阶段，清理各种已经分配的资源，同时保存仿真过程中产生的数据。图 3-2 所示是 Simulink 仿真模块的流程图。

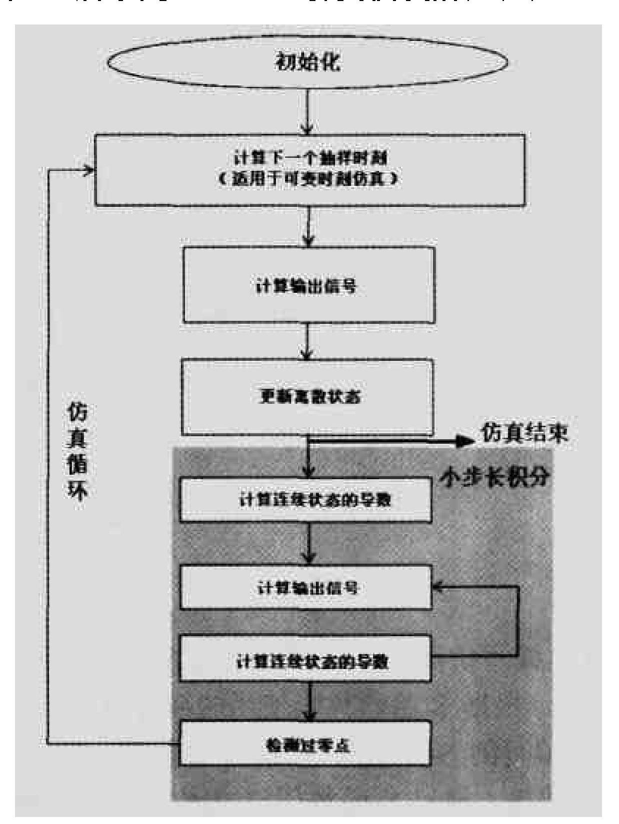



图 3-2 Simulink 模块地仿真流程

对应于仿真流程中的每一个步骤，Simulink 中的 S-函数调用预先设定的函数来实现相应的功能。例如，我们可以编写一个 mdlInitializeSizes 函数实现 S-函数的初始化操作，通过 mdlDerivatives 和 mdlUpdate 函数在每一个抽样时刻分别计算连续状态变量的导数和更新离散状态的数值，在 mdlOutputs 函数中计算 S-函数的输出信号等。需要指出的是，这些函数的名称都可以由用户自己设定。用户需要在 S-函数的主体部分对这些函数进行注册，Simulink 通过回调函数（Callback Function）的方式在不同事件发生的时候调用相应的函数。

## 3.1.2 S-函数基本概念

在编写 S-函数的时候经常涉及到的概念有 3 个：直接反馈（Direct feedthrough）、可变长度输入（Dynamically sized inputs）以及抽样时刻和偏移（Setting sample times and offsets），它们是编写 S-函数的基础。

### 1. 直接反馈

简单说来，直接反馈（Direct feedthrough）指的是输入信号是否直接影响着输出信号和

仿真的抽样时间。在计算 S-函数输出信号的过程中，如果输出信号（包括仿真过程中绘制的图形）是输入信号的函数，那么这个 S-函数存在直接反馈。同样地，如果在可变步长的仿真过程中，S-函数的输入信号影响着对下一个仿真时刻的计算，这个 S-函数也存在直接反馈。

Simulink 根据模块中的 S-函数是否存在直接反馈来确定仿真模型中各个模块的执行顺序。如果 S-函数不存在直接反馈，在计算该模块的输出信号时这个模块就可以不等待前一个模块的输出信号，因而有可能先于前面的模块执行。在编写 S-函数的过程中，首要的是确定本模块的直接反馈类型。

### 2. 可变长度输入

S-函数输入信号的长度既可以是固定的，也可以在仿真过程中根据输入信号的长度动态设定。同时，输入信号长度的动态变化也影响着 S-函数的连续状态、离散状态以及输出信号的长度，从而给 S-函数的设计提供了很大的灵活性。

对于 M 文件 S-函数构成的模块，它只有一个输入信号端口，只能接受一维输入向量，但是这个输入向量的长度可以是动态确定的。C 语言 S-函数则可以有多个输入、输出端口，每个端口的长度都是可变的。

如果 S-函数把它的连续状态、离散状态或输出信号也设置为可变长度信号，Simulink 将根据输入信号的长度来确定它们的长度，并且这些信号的长度与输入信号的长度相同。

### 3. 抽样时刻和偏移

在 Simulink 中，由 S-函数构造的模块的抽样时间既可以是固定的，也可以是连续的；既可以是连续的，也可以是离散的。抽样时间一般表示成“sample_time, offset_time”的形式，其中 sample_time 表示抽样周期，offset_time 表示每个抽样周期内的时间偏移。归纳起来，S-函数模块的抽样时间有以下几种：

■ 连续抽样时间（Continuous sample time）

连续抽样时间方式适用于模块中的 S-函数设置了连续状态，或者是模块中存在着未被检测到的过零点。这时候 Simulink 将调整仿真步长，采用小步长方式（Minor Step）进行积分运算。

连续抽样时间一般可以表示成二元组的形式[CONTINUOUS_SAMPLE_TIME, 0.0]，其中 CONTINUOUS_SAMPLE_TIME 是个常数，且 CONTINUOUS_SAMPLE_TIME = 0.0。

■ 连续时间固定间隔（Continuous but fixed in minor time step sample time）

如果模块中的 S-函数中包含连续状态，但是这些连续状态在小步长间隔内不改变数值，这时候可以采用连续时间固定间隔模式。

连续时间固定间隔一般可以表示成二元组的形式[CONTINUOUS_SAMPLE_TIME, FIXED_IN_MINOR_STEP_OFFSET]，二元组中的元素 CONTINUOUS_SAMPLE_TIME 和 FIXED_IN_MINOR_STEP_OFFSET 都是常数，且 FIXED_IN_MINOR_STEP_OFFSET = 1.0。

■ 离散抽样时间（Discrete sample time）

如果模块中的 S-函数只有离散状态，这时候 Simulink 可以采用离散抽样时间方式，以固定的周期执行仿真。

离散抽样间隔一般可以表示成二元组的形式[discrete_sample_time_period, offset]，其中 discrete_sample_time_period 表示抽样周期，offset 表示每个抽样周期内的时间偏移，且

0 ≤ offest<discrete sample time period 。在 这 种 模 式 下， Simul

返回相应的结果。

### 1. 输入和输出

M 文件 S-函数输入信号的个数是可变的，但是前 3 个参数 *t* 、*x*、*u* 以及 *flag* 是必不可少的，它们分别表示仿真时间、仿真模块的内部状态、仿真的输入信号以及所需执行的操作。另外，在仿真过程中，Simulink 还可以向 S-函数传递一些额外的参数 *p1*、*p2*…，它们可以用于 S-函数的各个计算过程中。

对应于 *flag* 的不同设置，S-函数执行不同的操作，从而向 Simulink 返回不同的结果。表 3-1 列出了 *flag* 参数的含义。

**表 3-1　　参数 flag 的含义**

| flag | S-函数名称 | 说明 |
|---|---|---|
| 0 | mdlInitializeSizes | S-函数的初始化，包括抽样间隔、连续状态和离散状态的初始设置等 |
| 1 | mdlDerivatives | 计算连续状态变量的导数 |
| 2 | mdlUpdate | 更新离散状态的数值 |
| 3 | mdlOutputs | 计算 S-函数的输出 |
| 4 | mdlGetTimeOfNextVarHit | 计算下一次仿真的时间，只适用于可变步长仿真 |
| 9 | mdlTerminate | 结束仿真 |

S-函数的输出信号包含 4 个元素，其中 *sys* 是 S-函数的计算结果，它的含义取决于 *flag* 的取值。例如，当 *flag* 等于 3 时，*sys* 表示 S-函数的输出信号。*x0* 表示 S-函数模块的初始状态。当 *flag* 等于 0 时，*x0* 返回 S-函数中各个状态的初始设置；当 *flag* 不等于 0 时本参数被忽略。如果 S-函数中没有状态，则 *x0* 是一个空向量。*str* 暂时保留，应该设置为空向量[]。*ts* 是一个具有两列元素的矩阵，其中第一列元素表示抽样时间，第二列元素表示时间偏移。

### 2. M 文件 S-函数模板

Simulink 提供了一个 M 文件 S-函数模板 sfuntmpl.m（该文件存放在 MATLAB 安装目录下的/toolbox/simulink/blocks 中），它提供了 M 文件 S-函数的基本构架，通过它可以快速地编写函数代码。为了对 M 文件 S-函数有一个基本的认识，我们结合 M 文件 S-函数模板的代码来说明 M 文件 S-函数的构成。下面的程序段列出了 M 文件 S-函数模板 sfuntmpl.m 的代码（其中省略了若干注释行）。

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% M 文件 S-函数的主体部分
% 函数名称：sfuntmpl
% 主要功能：根据输入参数 flag 的数值调用相应的函数
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch flag,
```

```
  case 0,
    % 当 flag 等于 0 时调用 mdlInitializeSizes 函数执行初始化
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 1,
    % 当 flag 等于 1 时调用 mdlDerivatives 函数计算连续状态的数值
    sys=mdlDerivatives(t,x,u);
  case 2,
    % 当 flag 等于 2 时调用 mdlUpdate 函数计算离散状态的数值
    sys=mdlUpdate(t,x,u);
  case 3,
    % 当 flag 等于 3 时调用 mdlOutputs 函数计算输出信号的数值
    sys=mdlOutputs(t,x,u);
  case 4,
    % 当 flag 等于 4 时调用 mdlGetTimeOfNextVarHit 函数计算下一个抽样时刻
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    % 当 flag 等于 9 时调用 mdlTerminate 函数结束仿真
    sys=mdlTerminate(t,x,u);
  otherwise
    % 当 flag 等于其他数值时表示仿真过程出错
    error(['Unhandled flag = ',num2str(flag)]);
end
% end sfuntmpl
%
%=============================================================================
% mdlInitializeSizes
% S-函数的初始化
% 向 Simulink 返回 S-函数各种信号的长度、初始设置和抽样时间设置
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
% 调用 simsizes 获得一个用于存放长度信息的结构
% -1 表示动态确定的长度
sizes = simsizes;
% 设置连续状态的个数
sizes.NumContStates = 0;
% 设置离散状态的个数
sizes.NumDiscStates = 0;
% 设置输出信号的个数
sizes.NumOutputs = 0;
% 设置输入信号的个数
```

```
sizes.NumInputs = 0;
% 设置直接反馈的状态：
%   0 表示没有直接反馈
%   1 表示存在直接反馈
sizes.DirFeedthrough = 1;
% 设置抽样时间的个数（大于等于 1）
sizes.NumSampleTimes = 1;
% 通过 simsizes 把 sizes 结构返回给 sys
sys = simsizes(sizes);
%
% 设置 S-函数的初始状态 x0
x0   = [];
%
% 设置 S-函数的保留参数 str（应该设置为空向量[]）
str = [];
%
% 初始化抽样时间
ts   = [0 0];
% end mdlInitializeSizes
%
%=============================================================================
% mdlDerivatives
% 计算 S-函数连续状态的导数，返回给 Simulink 进行积分计算
%=============================================================================
function sys=mdlDerivatives(t,x,u)
% 计算 S-函数连续状态的导数并且通过 sys 参数返回给 Simulink
sys = [];
% end mdlDerivatives
%
%=============================================================================
% mdlUpdate
% 更新 S-函数的离散状态并且向 Simulink 返回这些状态的数值
%=============================================================================
function sys=mdlUpdate(t,x,u)
% 计算 S-函数的离散状态并且通过 sys 参数返回给 Simulink
sys = [];
% end mdlUpdate
%
%=============================================================================
% mdlOutputs
```

```
% 计算 S-函数的输出信号并且返回给 Simulink 作为模块的输出
%==============================================================================
function sys=mdlOutputs(t,x,u)
% 计算 S-函数的输出信号并且通过 sys 参数返回给 Simulink
sys = [];
% end mdlOutputs
%
%==============================================================================
% mdlGetTimeOfNextVarHit
% 计算 S-函数的下一个抽样时刻并且返回给 Simulink
% 本函数只用于可变步长离散时间仿真且在初始化过程中把 ts 设置为[-2 0]
%==============================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)
% 一个设置下一个抽样时刻的示例
% 下一个抽样时刻设置为与当前时刻相差 1 秒
sampleTime = 1;
sys = t + sampleTime;
% end mdlGetTimeOfNextVarHit
%
%==============================================================================
% mdlTerminate
% 在仿真结束时执行清理工作，如回收内存等
%==============================================================================
function sys=mdlTerminate(t,x,u)
% 设置返回参数 sys 为空矩阵[]
sys = [];
% end mdlTerminate
```

## 3.2.2 M 文件 S-函数的编写示例

学习编程的最简便和直接的方法就是读程序，编写 S-函数也不例外。在本节里，将通过 4 个示例程序来学习 M 文件 S-函数的设计方法。这 4 个示例程序均取自 MATLAB 自带的程序，内容涵盖离散状态系统、连续状态系统、混合状态系统以及可变仿真步长系统，不仅具有广泛的代表性，更重要的是这些程序简单明了，是 S-函数初学者的经典素材。每个示例程序都详细地介绍了程序的结构和功能，并且列出了相应的程序代码。同时，在程序段中还添加了若干注释，详细介绍各个代码段的功能。本书后面的各个章节也将提供若干个具有某种特定功能的 S-函数的实例，以加深读者对 S-函数的理解和掌握。

**1. 离散状态系统**

下面通过一个示例程序来说明离散状态系统的 M 文件 S-函数的编写方法，这个示例程

序是 MATLAB 安装目录下的\toolbox\simulink\simdemos\中的 sfcndemo_dsfunc.mdl 文件。图 3-3 所示是该模型的结构框图。

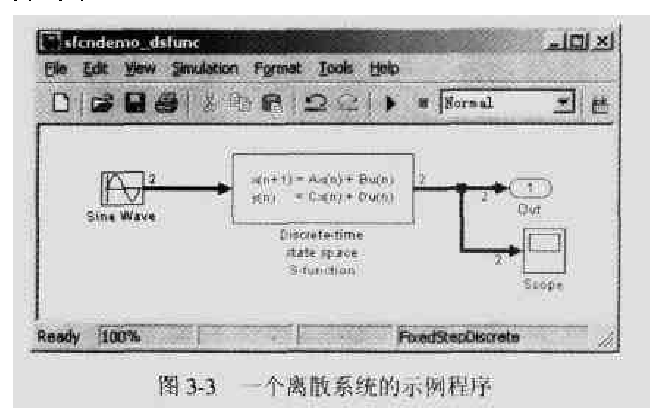



图 3-3　一个离散系统的示例程序

在这个示例程序中，正弦信号产生器（Sine Wave）产生一个具有两个元素的正弦信号向量，向量中的元素具有连续的抽样时间，信号幅度分别是 1 和 2。离散状态模块（Discrete-time state space S-function）对这个正弦信号进行变换，变换后的信号通过示波器模块（Scope）显示出来。表 3-2 所示是正弦信号产生器的参数设置。

**表 3-2　　正弦信号产生器（Sine Wave）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Sine Wave |
| Sine type | Time based |
| Amplitude | [1 2] |
| Bias | 0 |
| Frequency (rad/sec) | 1 |
| Phase (rad) | 0 |
| Sample time | 0 |
| Interpret vector parameters as 1-D | Checked |

同时，本示例程序采用离散时间且固定步长方式进行仿真。表 3-3 列出了仿真参数设置窗口中 Solver 面板的参数设置情况。

**表 3-3　　仿真参数设置**

| 参数名称 | 参数值 |
|---|---|
| Start time | 0.0 |
| Stop time | 10.0 |
| Type | Fixed-step<br>discrete (no continuous states) |
| Fixed step size | auto |
| Mode | auto |

离散状态模块（Discrete-time state space S-function）由 M 文件 S-函数 dsfunc.m 构造，这个 M 文件 S-函数位于 MATLAB 安装目录下的\toolbox\simulink\blocks 中。在使用 S-函数模块（S-Function）时，将参数 S-function name 设置为相应的 S-函数的名称，然后在 S-function parameters 中填写 S-函数的参数。在本示例程序中，S-函数的名称为 dsfunc，它没有额外的

参数，如图 3-4 所示。

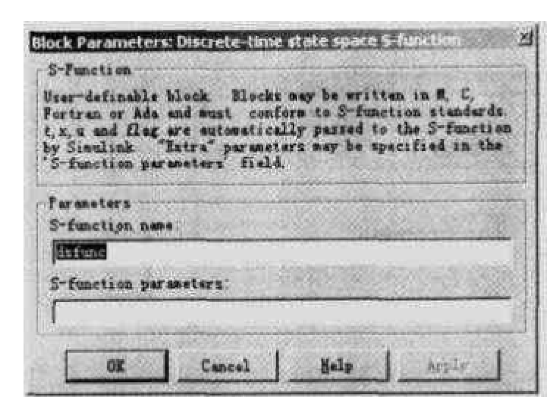



图 3-4 离散状态模块的参数设置

离散状态模块的功能是通过 M 文件 S-函数 dsfunc.m 实现的，它对输入信号向量实施矩阵变换：

$$\begin{cases} x(n+1) = Ax(n) + Bu(n) \\ y(n) = Cx(n) + Du(n) \end{cases},$$

其中变换矩阵：

$$A = \begin{pmatrix} -1.3839 & -0.5097 \\ 1.0000 & 0 \end{pmatrix},\quad B = \begin{pmatrix} -2.5559 & 0 \\ 0 & 4.2382 \end{pmatrix},\quad C = \begin{pmatrix} 0 & 2.0761 \\ 0 & 7.7891 \end{pmatrix},\quad D = \begin{pmatrix} -0.8141 & -2.9334 \\ 1.2426 & 0 \end{pmatrix}。$$

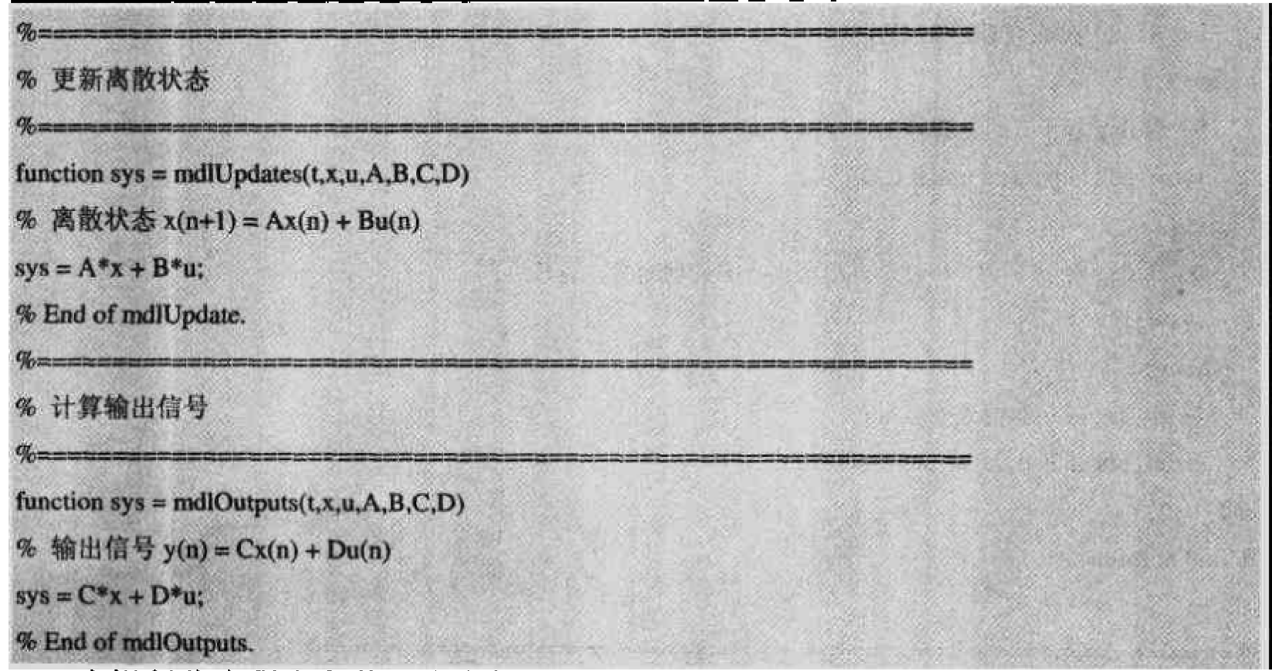
下面的程序段是 M 文件 S-函数 dsfunc.m 的代码。

```
function [sys,x0,str,ts] = dsfunc(t,x,u,flag)
% 本函数对输入信号向量实施矩阵变换
%       x(n+1) = Ax(n) + Bu(n)
%       y(n)   = Cx(n) + Du(n)
% 设置变换矩阵 A、B、C 和 D
A=[-1.3839    -0.5097
    1.0000          0];
B=[-2.5559          0
        0     4.2382];
C=[      0    2.0761
        0     7.7891];
D=[-0.8141    -2.9334
    1.2426          0];

switch flag,
  case 0
    % 当 flag 等于 0 时实施初始化
    sys = mdlInitializeSizes(A,B,C,D);
  case 2
    % 当 flag 等于 2 时更新离散状态
```

```
    sys = mdlUpdate(t,x,u,A,B,C,D);
  case 3
    % 当 flag 等于 3 时计算输出信号的数值
    sys = mdlOutputs(t,x,u,A,B,C,D);
  case {1, 4, 9}
    % 当 flag 等于 1、4、9 时没有定义相应的处理函数，直接返回
    sys = [];
  otherwise
    % 当 flag 等于其他数值时报错
    error(['unhandled flag = ',num2str(flag)]);
end
% End of dsfunc.

%================================================================
% 初始化
%================================================================
function [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D)
% 调用 simsizes 创建一个用于保存长度信息的结构
sizes = simsizes;
% 设置连续状态的个数为 0
sizes.NumContStates   = 0;
% 设置离散状态的个数为 2
sizes.NumDiscStates   = 2;
% 设置输出信号的个数为 2
sizes.NumOutputs      = 2;
% 设置输入信号的个数为 2
sizes.NumInputs       = 2;
% 本系统存在直接反馈
sizes.DirFeedthrough = 1;
% 设置抽样时间的个数为 1（即矩阵 ts 的行数）
sizes.NumSampleTimes = 1;
% 通过 sys 参数向 Simulink 返回长度信息
sys = simsizes(sizes);
% 两个离散状态的初始值设置为 1
x0   = ones(2,1);
% 设置 str 为空矩阵（str 未使用）
str = [];
% 设置抽样时间（抽样周期为 1 秒，偏移为 0）
ts   = [1 0];
% End of mdlInitializeSizes.
```



```
%=============================================================
% 更新离散状态
%=============================================================
function sys = mdlUpdates(t,x,u,A,B,C,D)
% 离散状态 x(n+1) = Ax(n) + Bu(n)
sys = A*x + B*u;
% End of mdlUpdate.
%=============================================================
% 计算输出信号
%=============================================================
function sys = mdlOutputs(t,x,u,A,B,C,D)
% 输出信号 y(n) = Cx(n) + Du(n)
sys = C*x + D*u;
% End of mdlOutputs.
```

在运行仿真程序之前，注意把 M 文件 S-函数文件（dsfunc.m）与仿真程序文件（sfcndemo_dsfunc.mdl）放置于相同目录下，否则，Simulink 将提示找不到相应的文件。图 3-5 所示是在 Simulink 中运行本示例程序之后示波器模块的输出信号。从图 3-5 所示中可以看到，M 文件 S-函数文件 dsfunc.m 的输出信号是周期为 1 的离散信号。

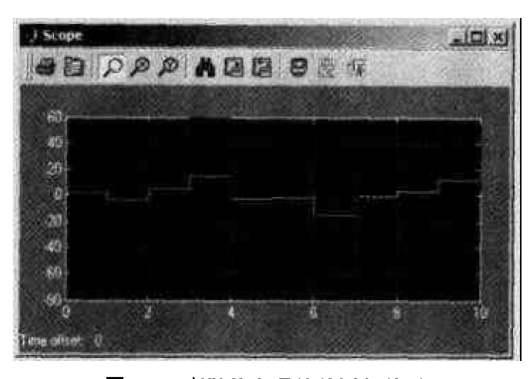



图 3-5　离散状态系统的运行结果

## 2. 连续状态系统

本节我们通过一个示例程序来说明离散状态系统的 M 文件 S-函数的编写方法，这个示例程序是 MATLAB 安装目录下的\toolbox\simulink\simdemos\中的 sfcndemo_csfunc.mdl 文件，图 3-6 所示是该模型的结构框图。

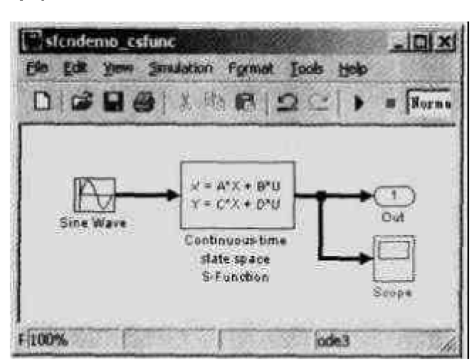



图 3-6　一个连续状态系统的示例程序

在这个示例程序中，正弦信号产生器（Sine Wave）产生一个具有两个元素的正弦信号向量，向量中的元素具有连续的抽样时间，信号幅度分别是 1 和 2。连续状态模块（Continuous-time state space S-function）对这个正弦信号进行变换，变换后的信号通过示波器模块（Scope）显示出来。正弦信号发生器的参数设置与上节中的设置相同，如表 3-4 所示。同时，本示例程序采用固定步长方式（间隔为 0.01 秒）进行仿真。表 3-4 列出了仿真参数设置窗口中 Solver 面板的参数设置情况。

**表 3-4　　仿真参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| Start time | 0.0 |
| Stop time | 10.0 |
| Type | Fixed-step<br>ode3 (Bogacki-Shampine) |
| Fixed step size | 0.01 |
| Mode | SingleTasking |

连续状态模块（Continuous-time state space S-function）由 M 文件 S-函数 csfunc.m 构造，这个 M 文件 S-函数位于 MATLAB 安装目录下的\toolbox\simulink\blocks。在本示例程序中，我们使用了 S-函数模块(S-Function)，其中 S-function name 设置为 csfunc，S-function parameters 为空，如图 3-7 所示。

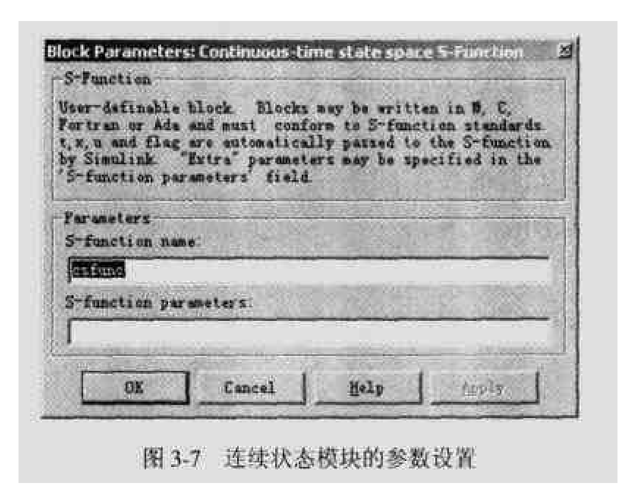



图 3-7　连续状态模块的参数设置

连续状态模块的功能是通过 M 文件 S-函数 csfunc.m 实现的，它对输入信号向量实施矩阵变换

$$\begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$$

其中变换矩阵

$$A = \begin{pmatrix} -0.09 & -0.01 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & -7 \\ 0 & -2 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 2 \\ 1 & -5 \end{pmatrix}, \quad D = \begin{pmatrix} -3 & 0 \\ 1 & 0 \end{pmatrix}$$

下面的程序段是 M 文件 S-函数 csfunc.m 的代码。

```
function [sys,x0,str,ts] = csfunc(t,x,u,flag)
```

```
% 本函数对输入信号向量实施矩阵变换
%       x' = Ax + Bu
%       y  = Cx + Du
%
% 设置矩阵变换的参数 A、B、C、D
A=[-0.09    -0.01
    1          0];
B=[ 1    -7
    0    -2];
C=[ 0     2
    1    -5];
D=[-3     0
    1     0];

switch flag,
  case 0
    % 当 flag 等于 0 时实施初始化
    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
  case 1
    % 当 flag 等于 1 时重新计算连续状态的导数
    sys = mdlDerivatives(t,x,u,A,B,C,D);
  case 3
    % 当 flag 等于 3 时计算输出信号的数值
    sys = mdlOutputs(t,x,u,A,B,C,D);
  case { 2, 4, 9 }
    % 当 flag 等于 2、4、9 时没有定义相应的处理函数，直接返回
    sys = [];
  otherwise
    % 当 flag 等于其他数值时报错
    error(['Unhandled flag = ',num2str(flag)]);
end
% End of csfunc.

%==============================================================
% 初始化
%==============================================================
function [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D)
% 调用 simsizes 创建一个用于保存长度信息的结构
sizes = simsizes;
% 设置连续状态的个数为 2
```

```
sizes.NumContStates  = 2;
% 设置离散状态的个数为 0
sizes.NumDiscStates  = 0;
% 设置输出信号的个数为 2
sizes.NumOutputs     = 2;
% 设置输入信号的个数为 2
sizes.NumInputs      = 2;
% 本系统存在直接反馈
sizes.DirFeedthrough = 1;
% 设置抽样时间的个数为 1（即矩阵 ts 的行数）
sizes.NumSampleTimes = 1;
% 通过 sys 参数向 Simulink 返回长度信息
sys = simsizes(sizes);
% 两个连续状态的初始值设置为 0
x0 = zeros(2,1);
% 设置 str 为空矩阵（str 未使用）
str = [];
% 设置抽样时间（[0 0]表示连续抽样时间）
ts = [0 0];
% End of mdlInitializeSizes.

%==============================================================
% 重新计算连续状态的导数
%==============================================================
function sys = mdlDerivatives(t,x,u,A,B,C,D)
% 连续状态 x' = Ax + Bu
sys = A*x + B*u;
% End of mdlDerivatives.

%==============================================================
% 计算输出信号
%==============================================================
function sys = mdlOutputs(t,x,u,A,B,C,D)
% 输出信号 y(n) = Cx(n) + Du(n)
sys = C*x + D*u;
% End of mdlOutputs.
```

把 M 文件 S-函数文件（csfunc.m）与仿真程序文件（sfcndemo_csfunc.mdl）放置于相同目录下，启动仿真程序，运行结束之后得到如图 3-8 所示的示波器信号。从图 3-8 所示中可以看到，M 文件 S-函数文件 csfunc.m 的输出信号是一个连续信号。

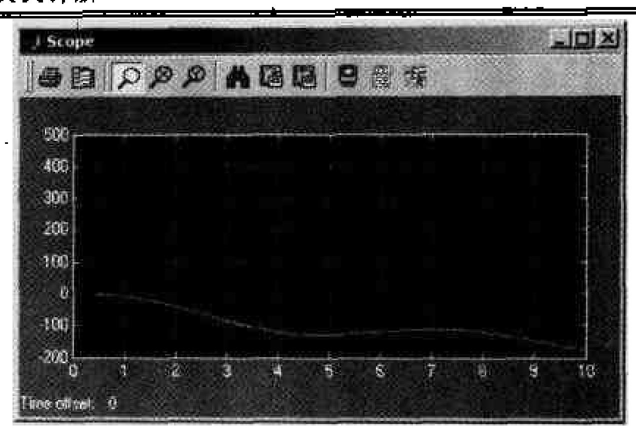


图 3-8 连续状态系统示例程序的运行结果

### 3. 混合系统

前面我们介绍了离散状态系统和连续状态系统的 M 文件 S-函数的设计方法，在同时具有离散状态和连续状态的混合系统中，M 文件 S-函数既要设计针对离散系统的状态更新函数，同时也要编写计算连续状态导数的代码。本节将介绍这种混合系统的 M 文件 S-函数的设计方法。

此例使用的混合系统由一个 Integrator（积分器）和一个 Unit Delay（单位时延模块）组成，输入信号首先通过积分器进行积分，然后通过单位延时模块把积分信号延时一个单位之后输出，如图 3-9 所示。

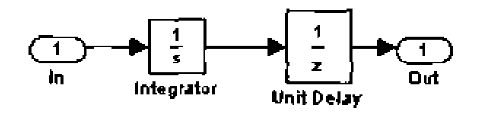



图 3-9 混合系统的构成

本节介绍的示例程序 sfcndemo_mixedm.mdl 将通过一个 S-函数模块来实现上述混合系统的功能，它存放在 MATLAB 安装目录下的\toolbox\simulink\simdemos 中。图 3-10 所示是该模型的结构框图。

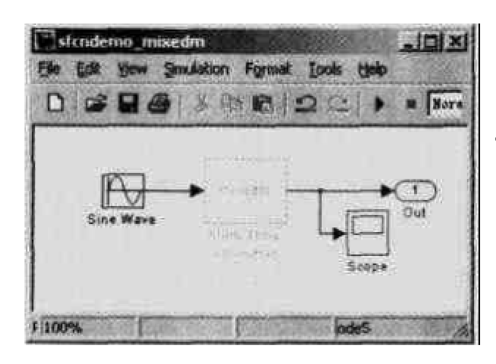



图 3-10 一个混合系统的示例

在这个示例程序中，Sine Wave（正弦信号产生器）产生一个幅度为 1 的正弦信号，然后通过 Multi-time S-function（混合状态模块）对这个正弦信号进行积分和延迟，最后进入 Scope（示波器模块）。表 3-5 所示是正弦信号产生器的参数设置。

表 3-5　　正弦信号产生器（Sine Wave）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Sine Wave |
| Sine type | Time based |
| Amplitude | 1 |
| Bias | 0 |
| Frequency (rad/sec) | 1 |
| Phase (rad) | 0 |
| Sample time | 0 |
| Interpret vector parameters as 1-D | Checked |

本示例程序采用固定步长方式进行仿真，抽样间隔等于 0.1 秒。表 3-6 列出了仿真参数设置窗口中 Solver 面板的参数设置情况。

表 3-6　　仿真参数设置

| 参数名称 | 参数值 |
|---|---|
| Start time | 0.0 |
| Stop time | 10.0 |
| Type | Fixed-step<br>ode5 (Dormand-Prince) |
| Fixed step size | 0.1 |
| Mode | SingleTasking |

混合状态模块（Multi-time S-function）由 M 文件 S-函数 mixedm.m 构造，这个 M 文件 S-函数位于 MATLAB 安装目录下的\toolbox\simulink\blocks。在使用 S-函数模块（S-Function）构造混合状态模块时，将参数 S-function name 设置为 mixedm，S-function parameters 设置为空。下面的程序段是 M 文件 S-函数 mixedm.m 的代码。

```
function [sys,x0,str,ts] = mixedm(t,x,u,flag)
% 本函数首先对输入信号实施积分，然后把积分信号单位延时
%
% 设置单位延时的周期和偏移
dperiod = 1;
doffset = 0;

switch flag,
  case 0
    % 当 flag 等于 0 时实施初始化
    [sys,x0,str,ts] = mdlInitializeSizes(dperiod,doffset);
  case 1
    % 当 flag 等于 1 时重新计算连续状态的导数
    sys = mdlDerivatives(t,x,u);
  case 2
```

```
    % 当 flag 等于 2 时更新离散状态
    sys = mdlUpdate(t,x,u,dperiod,doffset);
  case 3
    % 当 flag 等于 3 时计算输出信号
    sys = mdlOutputs(t,x,u,doffset,dperiod);
  case {4, 9}
    % 当 flag 等于 4、9 时没有定义相应的处理函数，直接返回
    sys = [];
  otherwise
    % 当 flag 等于其他数值时报错
    error(['unhandled flag = ',num2str(flag)]); % Error handling
end
% End of mixedm.
%==============================================================
% 初始化
%==============================================================
function [sys,x0,str,ts] = mdlInitializeSizes(dperiod,doffset)
% 调用 simsizes 创建一个用于保存长度信息的结构
sizes = simsizes;
% 设置连续状态的个数为 1
sizes.NumContStates  = 1;
% 设置离散状态的个数为 1
sizes.NumDiscStates  = 1;
% 设置输出信号的个数为 1
sizes.NumOutputs     = 1;
% 设置输入信号的个数为 1
sizes.NumInputs      = 1;
% 本系统不存在直接反馈
sizes.DirFeedthrough = 0;
% 设置抽样时间的个数为 2（即矩阵 ts 的行数）
sizes.NumSampleTimes = 2;
% 通过 sys 参数向 Simulink 返回长度信息
sys = simsizes(sizes);
% 两个状态（离散状态和连续状态）的初始值设置为 1
x0  = ones(2,1);
% 设置 str 为空矩阵（str 未使用）
str = [];
% 设置抽样时间
% 第一行元素[0 0]表示连续抽样时间
% 第二行元素[dperiod, doffset]表示间隔为 dperiod 的离散抽样时间
```

```
ts   = [0,          0;
        dperiod, doffset];
% End of mdlInitializeSizes.
%
%=============================================================
% 重新计算连续状态的导数
%=============================================================
function sys = mdlDerivatives(t,x,u)
% 连续状态 x' =u
sys = u;
% end of mdlDerivatives.
%=============================================================
% 更新离散状态
%=============================================================
function sys = mdlUpdate(t,x,u,dperiod,doffset)
% 在离散状态的抽样时刻到来时（或误差不超过 1e-8 时）更新离散状态
% 离散状态的数值等于连续状态的当前值
if abs(round((t-doffset)/dperiod)-(t-doffset)/dperiod) < 1e-8
  sys = x(1);
% 否则，返回一个空向量，使得 Simulink 能够保持离散状态的原数值
else
  sys = [];
end
% End of mdlUpdate.
%=============================================================
% 计算输出信号
%=============================================================
function sys = mdlOutputs(t,x,u,doffset,dperiod)
% 在离散状态的抽样时刻到来时（或误差不超过 1e-8 时）更新输出信号
% 输出信号的数值等于离散状态的当前值
if abs(round((t-doffset)/dperiod)-(t-doffset)/dperiod) <1e-8
  sys = x(2);
% 否则，保持原来的输出信号，从而引入一个单位时延
else
  sys = [];
end
% End of mdlOutputs.
```

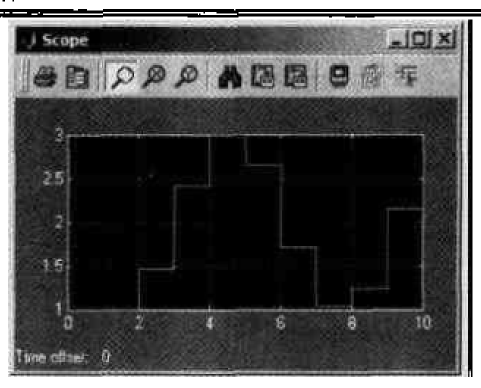



图 3-11　混合系统示例程序的运行结果

图 3-11 所示的仿真结束之后示波器的信号。从图 3-11 所示中可以看到，M 文件 S-函数文件 mixed.m 的输出信号是一个周期为 1 的离散信号，它是对连续的积分信号实施抽样和保持的结果。

### 4. 可变仿真步长系统

前面曾经说过，S-函数的抽样时刻可以是变化的，这时候 Simulink 通过调用相应的函数来确定下一个仿真时刻。本节通过一个 MATLAB 示例程序（sfcndemo_vsfunc.mdl）来说明可变仿真步长的 M 文件 S-函数的设计方法，它存放在 MATLAB 安装目录下的\toolbox\simulink\simdemos 中。图 3-12 所示是这个示例程序的系统框图，Continuous-Time Variable Step S-function（可变步长模块）的输入信号是由两个信号复用而成的，其中第一个输入信号是一个正弦信号，第二个输入信号则是一个截掉后的正弦信号，它是由一个幅度为 2 的正弦信号通过一个 Switch（选通器模块）之后得到的。

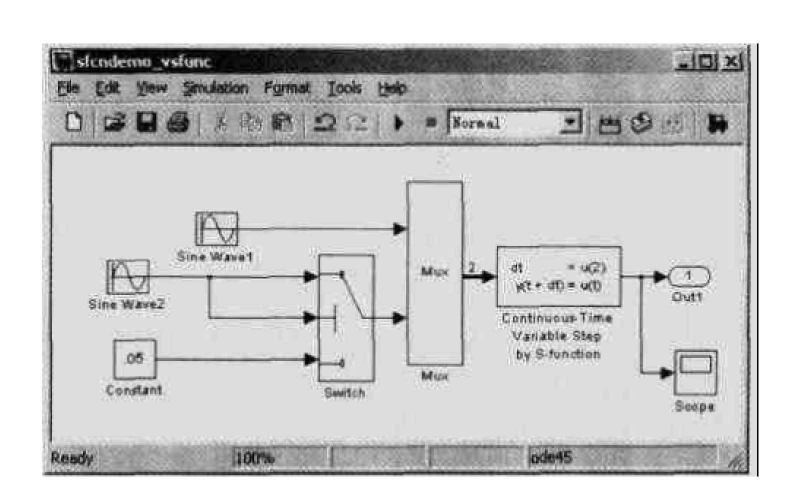



图 3-12　一个可变仿真步长系统的示例

在这个示例程序中，两个正弦信号产生器（Sine Wave1 和 Sine Wave2）产生一个幅度分别为 1 和 2 的正弦信号，表 3-7 所示是正弦信号产生器的参数设置。第二个正弦信号发生器产生的正弦信号通过 Switch（选通器模块），这个选通模块有 3 个输入端口，当第二个输入端口的输入信号大于一个预先设定的数值时，选通器输出第一个输入端口的输入信号；否则，选通器的输出信号等于第三个输入端口的输入信号。正弦信号产生器模块和选通器模块的参数设置分别如表 3-7 和表 3-8 所示。

表 3-7 **Sine Wave（正弦信号产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Sine Wave |
| Sine type | Time based |
| Amplitude | 1 (Sine Wave1 模块)<br>2 (Sine Wave2 模块) |
| Bias | 0 |
| Frequency (rad/sec) | 2*pi |
| Phase (rad) | 0 |
| Sample time | 0 |
| Interpret vector parameters as 1-D | Checked |

表 3-8 **Switch（选通器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Switch |
| Criteria for passing first input | u2 >= Threshold |
| Threshold | 0.1 |

第一个正弦信号产生器和选通器模块的输出信号通过 Mux（复用器）复合成一个信号进入 Continuous-Time Variable Step S-function（可变步长模块），产生的输出信号通过示波器模块（Scope）显示出来。本示例程序采用可变步长方式进行仿真，仿真参数设置窗口中 Solver 面板的参数设置情况如表 3-9 所示。

表 3-9 **仿真参数设置**

| 参数名称 | 参数值 |
|---|---|
| Start time | 0.0 |
| Stop time | 50.0 |
| Type | Variable-step<br>ode45 (Dormand-Prince) |
| Max step size | auto |
| Min step size | auto |
| Initial step size | auto |
| Relative tolerance | 1e-3 |
| Absolute tolerance | auto |
| Output options | Refine output |
| Refine factor | 1 |

可变步长模块由 M 文件 S-函数 vsfunc.m 构造，这个 M 文件 S-函数位于 MATLAB 安装目录下的\toolbox\simulink\blocks。在使用 S-函数模块（S-Function）构造混合状态模块时，将参数 S-function name 设置为 vsfunc，S-function parameters 设置为空，下面的程序段是 M 文件 S-函数 vsfunc.m 的代码。

```
function [sys,x0,str,ts] = vsfunc(t,x,u,flag)
```

```
% 本函数对第一个输入端口的输入信号进行延迟输出
% 延迟时间的长度由第二个输入端口的输入信号确定
%
%       dt          = u(2)
%       y(t+dt) = u(t)
switch flag,
  case 0
    % 当 flag 等于 0 时实施初始化
    [sys,x0,str,ts] = mdlInitializeSizes;
  case 2
    % 当 flag 等于 2 时更新离散状态
    sys = mdlUpdate(t,x,u);
  case 3
    % 当 flag 等于 3 时计算输出信号
    sys = mdlOutputs(t,x,u); % Calculate outputs
  case 4
    % 当 flag 等于 4 时计算 S-函数的下一个仿真时刻
    sys = mdlGetTimeOfNextVarHit(t,x,u);
  case { 1, 9 }
    % 当 flag 等于 1、9 时没有定义相应的处理函数，直接返回
    sys = [];
  otherwise
    % 当 flag 等于其他数值时报错
    error(['Unhandled flag = ',num2str(flag)]);
end
% End of vsfunc.
%=============================================================
% 初始化
%=============================================================
function [sys,x0,str,ts] = mdlInitializeSizes
% 调用 simsizes 创建一个用于保存长度信息的结构
sizes = simsizes;
% 设置连续状态的个数为 0
sizes.NumContStates    = 0;
% 设置离散状态的个数为 1
sizes.NumDiscStates    = 1;
% 设置输出信号的个数为 1
sizes.NumOutputs        = 1;
% 设置输入信号的个数为 2
sizes.NumInputs         = 2;
```

```
% 本系统存在直接反馈（抽样时间由第二个输入信号决定）
sizes.DirFeedthrough = 1;
% 设置抽样时间的个数为 1（即矩阵 ts 的行数）
sizes.NumSampleTimes = 1;
% 通过 sys 参数向 Simulink 返回长度信息
sys = simsizes(sizes);
% 离散状态的初始值设置为 0
x0 = [0];
% 设置 str 为空矩阵（str 未使用）
str = [];
% 设置抽样时间（[-2 0]表示可变仿真步长）
ts = [-2 0];
% End of mdlInitializeSizes.
%==============================================================
% 更新离散状态
%==============================================================
function sys = mdlUpdate(t,x,u)
% 离散状态的当前值等于第一个输入端口的输入信号
sys = u(1);
% End of mdlUpdate.
%==============================================================
% 计算输出信号
%==============================================================
function sys = mdlOutputs(t,x,u)
%输出信号等于离散状态的当前值
sys = x(1);
% end mdlOutputs
%
%==============================================================
% 计算 S-函数的下一个仿真时刻
%==============================================================
function sys = mdlGetTimeOfNextVarHit(t,x,u)
% S-函数的下一个仿真时刻等于当前时刻与第二个输入端口的输入信号之和
sys = t + u(2);
% End of mdlGetTimeOfNextVarHit.
```

图 3-13 所示是本示例程序的运行结果。从图 3-13 所示中可以看到一些比较尖锐的信号幅度变化，这是由于第二个正弦信号的幅度低于 0.1 时，选通器的输出信号等于 0.05，可变步长模块的抽样周期变成 0.05 秒，从而造成比较尖锐的幅度变化。当第二个正弦信号的幅度大于 0.1 时，选通器的输出信号就等于这个正弦信号的幅度，这时候可变步长模块的抽样周期取决于第二个正弦信号的幅度。

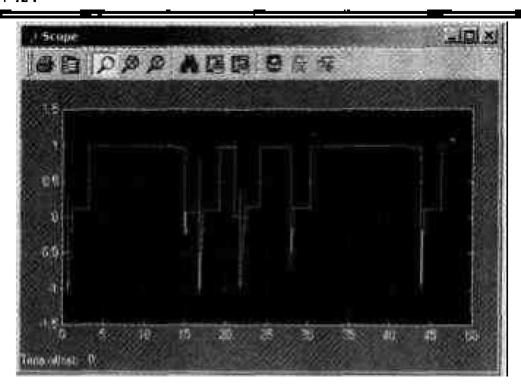


图 3-13 可变仿真步长系统的运行结果

# 3.3 C 语言 S-函数

C 语言 S-函数的代码采用通用的 C 语言规范，因而同时具有 C 语言和 S-函数的特点，更适合于设计通用的 S-函数模块。在本节里，我们将首先介绍 C 语言 S-函数的特点和 C 语言 S-函数模板，然后通过几个示例程序来说明 C 语言 S-函数的编写方法。

## 3.3.1 C 语言 S-函数简介

C 语言 S-函数是采用 C 语言编写的一种 S-函数，因此它具有 S-函数的一般特性。C 语言 S-函数的一个重要特性是它具有比 M 文件 S-函数更强的处理能力，能够更好地实现对仿真过程的控制。另外，通过 Simulink 中的 S-函数构造模块（S-Function Builder）可以快速地生成 C 语言 S-函数的代码。

**1. C 语言 S-函数的特点**

C 语言 S-函数与 M 文件 S-函数相似，它采用回调函数的方式，根据 Simulink 的不同指令调用不同的函数（如初始化，更新离散状态，计算连续状态的导数，以及产生输出信号等等），然后向 Simulink 返回相应的结果。C 语言 S-函数编译之后产生 MATLAB 可执行文件（MEX，MATLAB EXecutable），这种可执行文件在不同的操作系统平台上有不同的方式，对于 Windows 平台，它产生动态连接库文件（DLL，Dynamic-Link Library）。

与 M 文件 S-函数相比，C 语言 S-函数能够提供更多的回调函数，从而具有更强的处理能力。我们知道，对于仿真过程中的每一个模块，Simulink 都为之建立一种称为 SimStruct 的数据结构，用于保存该模块的相关信息。这个结构的内容在 M 文件 S-函数中是不能读取的，但是在 C 语言 S-函数中，我们能够读取和修改 SimStruct 结构中的内容，因此，通过 C 语言 S-函数，我们能够实现对仿真过程更强的控制功能。

另外，C 语言 S-函数能够处理更多类型的数据，如矩阵信号等。为此，Simulink 为 C 语言 S-函数定义了更多的回调函数，用于实现更多的数据处理功能。在编写 C 语言 S-函数的过程中，我们不需要对所有这些回调函数进行定义。实际应用中我们通常根据模块需要实现的功能选择相应的回调函数。

C 语言 S-函数的代码一般由 3 部分组成：头部定义、函数体以及尾部定义。在头部定义

中，C 语言 S-函数通过定义宏来设置 C 语言 S-函数的名称，其一般格式如下：

```
#define S_FUNCTION_NAME   your_sfunction_name_here
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
```

函数体部分则对各种回调函数进行定义，如 mdlInitializeSizes 实现函数的初始化操作，mdlTerminate 在仿真结束时执行清理工作，等等。C 语言 S-函数中的回调函数的名称都是固定的，并且都带有一个 SimStruct 参数。

```
static void mdlInitializeSizes(SimStruct *S)
{
}
  .
<additional S-function routines/code>

static void mdlTerminate(SimStruct *S)
{
}
```

尾部定义则是对 C 语言 S-函数编译和链接功能的一些参数设置。如果需要把 C 语言 S-函数编译成 MEX 文件，Simulink 需要链接 MEX 接口文件 simulink.c，其代码如下：

```
#ifdef MATLAB_MEX_FILE      /* Is this file being compiled as a MEX-file? */
#include "simulink.c"       /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"        /* Code generation registration function */
#endif
```

创建 C 语言 S-函数的一种简便方法是使用 Simulink 中的 S-函数构造模块（S-Function Builder）。S-函数构造模块根据用户的各种设置自动生成 C 语言 S-函数的框架和代码，通过 S-函数构造模块，用户不需要重新构造函数的框架，但是 S-函数构造模块在一定程度上限制了 C 语言 S-函数的功能，因为通过 S-函数构造模块构造的 S-函数只能有一个输入信号和一个输出信号，而且只能处理 double 类型的数据。

### 2. C 语言 S-函数模板

与 M 文件 S-函数类似地，Simulink 提供了一个 C 语言 S-函数模板 sfuntmpl_basic.c，这个文件存放在 MATLAB 安装目录下的 simulink/src 中。C 语言 S-函数模板提供了一个用 C 语言编写 S-函数的基本框架，通过它可以快速地编写函数代码。下面我们结合 C 语言 S-函数模板的代码来说明 C 语言 S-函数的基本结构。

```
/* sfuntmpl_basic.c C 语言 S-函数模板 */

/* 定义 S-函数的名称，用自己的名称替换 sfuntmpl_basic */
#define S_FUNCTION_NAME   sfuntmpl_basic
/* S-函数的版本，必须设置为 2 */
#define S_FUNCTION_LEVEL 2
```

```
/* 文件 simstruc.h 定义了 SimStruct 结构及相关的宏 */
#include "simstruc.h"

/* Function: mdlInitializeSizes ==============================================*/
/* S-函数的初始化                                                            */
static void mdlInitializeSizes(SimStruct *S)
{
    /* 设置 S-函数参数的个数（用参数数目替代 ssSetNumSFcnParams 中的 0） */
    ssSetNumSFcnParams(S, 0);
    /* 检查 S-函数参数数目是否与设置的数值一致 */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* 如果不一致则直接返回 */
        return;
    }

    /* 设置连续状态的数目 */
    ssSetNumContStates(S, 0);
    /* 设置离散状态的数目 */
    ssSetNumDiscStates(S, 0);

    /* 设置 S-函数输入端口的个数 */
    if (!ssSetNumInputPorts(S, 1)) return;
    /* 设置各个输入端口的宽度 */
    ssSetInputPortWidth(S, 0, 1);
    /* 设置各个输入端口中的元素是否存放在连续的内存中 */
    ssSetInputPortRequiredContiguous(S, 0, true);
    /* 设置 S-函数是否存在直接反馈 */
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    /* 设置 S-函数输出端口的个数 */
    if (!ssSetNumOutputPorts(S, 1)) return;
    /* 设置各个输出端口的宽度 */
    ssSetOutputPortWidth(S, 0, 1);
    /* 设置抽样时间的个数（抽样时间在 mdlInitializeSampleTimes 中进行定义） */
    ssSetNumSampleTimes(S, 1);
    /* 设置浮点数工作向量的长度 */
    ssSetNumRWork(S, 0);
    /* 设置整数工作向量的长度 */
    ssSetNumIWork(S, 0);
    /* 设置指针工作向量的长度 */
    ssSetNumPWork(S, 0);
```

```
    /* 设置工作模式向量的长度 */
    ssSetNumModes(S, 0);
    /* 设置过零点检测状态的个数 */
    ssSetNumNonsampledZCs(S, 0);
    /* 设置 S-函数工作模式的选项 */
    ssSetOptions(S, 0);
}

/* Function: mdlInitializeSampleTimes ========================================*/
/* 设置抽样时间                                                                 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* 设置抽样时间 */
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    /* 设置抽样时间偏移 */
    ssSetOffsetTime(S, 0, 0.0);
}

/* 如果不需要定义 mdlInitializeConditions 函数则将#define 设置为#undef */
#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
  /* Function: mdlInitializeConditions ========================================*/
  /* 初始化连续状态和离散状态                                                  */
  /* 通过 ssGetContStates(S)获得连续状态                                        */
  /* 通过 ssGetRealDiscStates(S)获得离散状态                                    */
  static void mdlInitializeConditions(SimStruct *S)
  {
  }
#endif /* MDL_INITIALIZE_CONDITIONS */

/* 如果不需要定义 mdlStart 函数则将#define 设置为#undef */
#define MDL_START
#if defined(MDL_START)
  /* Function: mdlStart =======================================================*/
  /* 仿真开始时的初始化操作                                                     */
  /* 在整个仿真过程中只执行一次                                                 */
  static void mdlStart(SimStruct *S)
  {
  }
#endif /*   MDL_START */
```

```
/* Function: mdlOutputs ===============================================*/
/* 计算输出信号的数值                                                  */
/* 通过 ssGetY(S)获得输出信号向量                                      */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    /* 获得输入端口的输入信号向量 */
    const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    /* 获得输出端口的输出信号向量 */
    real_T        *y = ssGetOutputPortSignal(S,0);
    /* 设置输出信号的数值（在这里把输出信号设置为输入信号） */
    y[0] = u[0];
}

/* 如果不需要定义 mdlUpdate 函数则将#define 设置为#undef */
#define MDL_UPDATE
#if defined(MDL_UPDATE)
  /* Function: mdlUpdate ==============================================*/
  /* 更新离散状态                                                      */
  /* 每个抽样时刻到来时执行一次                                        */
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */

/* 如果不需要定义 mdlDerivatives 函数则将#define 设置为#undef */
#define MDL_DERIVATIVES   /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
  /* Function: mdlDerivatives =========================================*/
  /* 计算连续状态的导数                                                */
  /* 通过 ssGetdX(S)获得连续状态导数向量                               */
  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */

/* Function: mdlTerminate ============================================*/
/* 执行仿真结束时的清理工作                                          */
static void mdlTerminate(SimStruct *S)
{
}
```

```
/* 如果本函数编译成 MEX 文件则链接 simulink.c 文件 */
/* 否则，链接 cg_sfun.h 文件                    */
#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

## 3.3.2　C 语言 S-函数的编写示例

C 语言 S-函数代码的编写在很大程度上与 M 文件 S-函数相似。本节将详细地讲解两个示例程序。这两个示例程序选自 MATLAB 提供的 S-函数实例，分别说明了无状态系统以及离散状态系统中 C 语言 S-函数的代码编写过程。

### 1. 无状态系统

本节通过一个简单的示例程序（sfcndemo_timestwo.mdl）来说明如何使用 C 语言 S-函数设计一个无状态系统。图 3-14 所示是该系统的结构框图。在这个示例程序中，Sine Wave（正弦信号产生器）产生一个幅度为 1 的正弦信号，然后通过一个 S-function（S-函数模块）把这个正弦信号执行乘 2 操作，最后这两个信号通过 Mux（复用模块）复用后输出到示波器模块（Scope）。

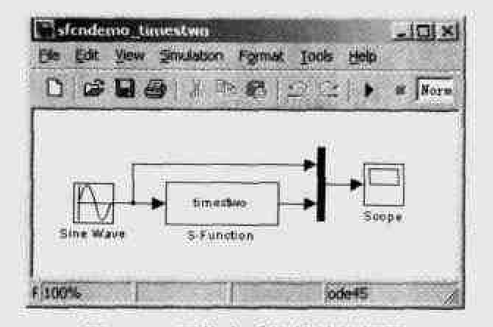



图 3-14　无状态系统的结构框图

图 3-15 所示是示波器的输出信号波形，其中黄色线条表示正弦信号产生器的输出信号，红色线条则是经过 S-函数模块变换之后的信号。在仿真过程中，我们使用了仿真参数的缺省设置，采用可变步长对模型进行仿真。从图 3-15 所示中可以看到，这个信号也是一个正弦信号，且幅度是原信号的两倍。

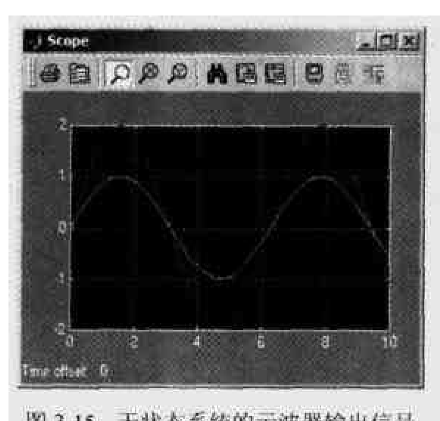



图 3-15　无状态系统的示波器输出信号

本示例程序中的 S-function（S-函数模块）是一个 C 语言 S-函数，其名称为 timestwo，并且不带有任何参数．这个 C 语言 S-函数存放在 MATLAB 安装目录下的\simulink\source 中，其功能是获取输入信号的数值，把这个数值乘于 2 后产生输出．下面我们来看看 timestwo.c 文件的代码。

```
/* timestwo.c  输出信号是输入信号的两倍      */
/*              y   = 2*u                     */

/* 定义 S-函数的名称为 timestwo */
#define S_FUNCTION_NAME   timestwo
/* S-函数的版本，必须设置为 2 */
#define S_FUNCTION_LEVEL 2
/* 文件 simstruc.h 定义了 SimStruct 结构及相关的宏 */
#include "simstruc.h"

/* Function: mdlInitializeSizes ============================================*/
/* S-函数的初始化                                                           */
static void mdlInitializeSizes(SimStruct *S)
{
    /* 设置 S-函数参数的个数为 0 */
    ssSetNumSFcnParams(S, 0);
    /* 检查 S-函数参数数目是否与设置的数值一致 */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* 如果不一致则直接返回 */
        return;
    }

    /* 设置 S-函数输入端口的个数为 1 */
    if (!ssSetNumInputPorts(S, 1)) return;
    /* 设置输入端口的宽度为动态确定 */
    ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
    /* 设置 S-函数存在直接反馈 */
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    /* 设置 S-函数输出端口的个数为 1 */
    if (!ssSetNumOutputPorts(S,1)) return;
    /* 设置输出端口的宽度为动态确定 */
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);

    /* 设置抽样时间的个数为 1（抽样时间在 mdlInitializeSampleTimes 中进行定义） */
    ssSetNumSampleTimes(S, 1);
```

```
    /* 设置S-函数工作模式的选项 */
    ssSetOptions(S,
                 SS_OPTION_WORKS_WITH_CODE_REUSE |
                 SS_OPTION_EXCEPTION_FREE_CODE |
                 SS_OPTION_USE_TLC_WITH_ACCELERATOR);
}

/* Function: mdlInitializeSampleTimes ==========================================*/
/* 设置抽样时间                                                                  */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* 设置抽样时间为继承方式（与输入信号相同） */
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    /* 设置抽样时间偏移为0 */
    ssSetOffsetTime(S, 0, 0.0);
}

/* Function: mdlOutputs ========================================================*/
/* 计算输出信号的数值 y = 2*u                                                    */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T             i;
    /* 获得指向输入信号指针向量的指针 */
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    /* 获得指向输出端口信号的指针 */
    real_T            *y     = ssGetOutputPortRealSignal(S,0);
    /* 获得输出端口信号的宽度 */
    int_T             width = ssGetOutputPortWidth(S,0);

    /* 计算输出信号中每个元素的数值 */
    for (i=0; i<width; i++) {
        /* 计算输出信号（输出信号等于输入信号的两倍） */
        *y++ = 2.0 *(*uPtrs[i]);
    }
}

/* Function: mdlTerminate ======================================================*/
/* 执行仿真结束时的清理工作                                                      */
/* 不需要执行任何操作，但是不能省略                                              */
static void mdlTerminate(SimStruct *S)
```

```
{
}

/* 如果本函数编译成 MEX 文件则链接 simulink.c 文件 */
/* 否则，链接 cg_sfun.h 文件                      */
#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

在完成 timestwo.c 文件的编写之后，需要对这个文件进行编译。MATLAB 的文件编译命令是 mex，因此，在工作区中输入命令行“mex timestwo.c”，如果编译过程中没有检测到代码错误，Simulink 产生一个 MEX 文件 timestwo.dll，这时候就可以执行 sfcndemo_timestwo 的仿真了。

## 2. 离散状态系统

本节介绍离散状态系统 C 语言 S-函数的编写方法，使用的示例程序为 MATLAB 安装目录下的\toolbox\simulink\simdemos\中的 sfcndemo_sfun_dynsize.mdl 文件。图 3-16 所示是实例 sfcndemo_sfun_dynsize 的结构框图。

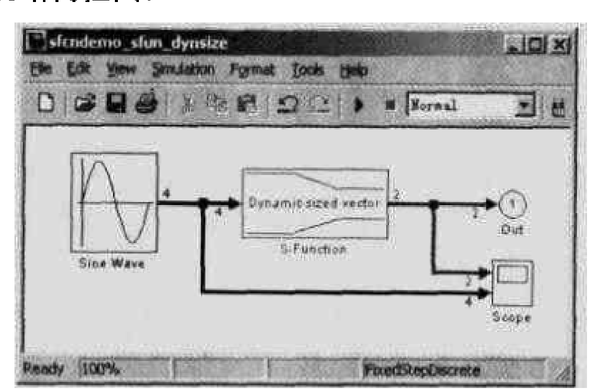



图 3-16 离散状态系统的结构框图

在 sfcndemo_sfun_dynsize 仿真模型中，Sine Wave（正弦信号产生器）产生一个具有 4 个元素的正弦信号向量，向量中的元素具有连续的抽样时间，信号幅度分别是 1、2、3 和 4。这些信号通过一个 S-function（S-函数模块）之后产生长度为 2 的输出向量，其中每个信号等于两个输入信号的和。最后，Scope（示波器模块）显示了 S-函数模块的输入信号和输出信号。正弦信号发生器的参数设置如表 3-10 所示。

表 3-10 Sine Wave（正弦信号产生器）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Sine Wave |
| Sine type | Time based |
| Amplitude | 1:4 |
| Bias | 0 |

续表

| 参数名称 | 参数值 |
| --- | --- |
| Frequency (rad/sec) | 1 |
| Phase (rad) | 0 |
| Sample time | 0 |
| Interpret vector parameters as 1-D | Checked |

本示例程序采用离散时间固定步长方式进行仿真，其仿真参数设置窗口中 Solver 面板的参数设置情况如表 3-11 所示。

**表 3-11 仿真参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| Start time | 0.0 |
| Stop time | 10.0 |
| Type | Fixed-step<br>discrete (no continuous states) |
| Fixed step size | auto |
| Mode | auto |

图 3-17 所示是 sfcndemo_sfun_dynsize.mdl 仿真结束之后示波器的输出波形。从图中可以看到，输入信号是长度为 4 的正弦信号（正弦信号的幅度分别等于 1、2、3 和 4），而输出信号则是一个长度为 2 的离散正弦信号，其幅度分别等于 3 和 7，即分别等于前两个输入信号以及后两个输入信号的和。

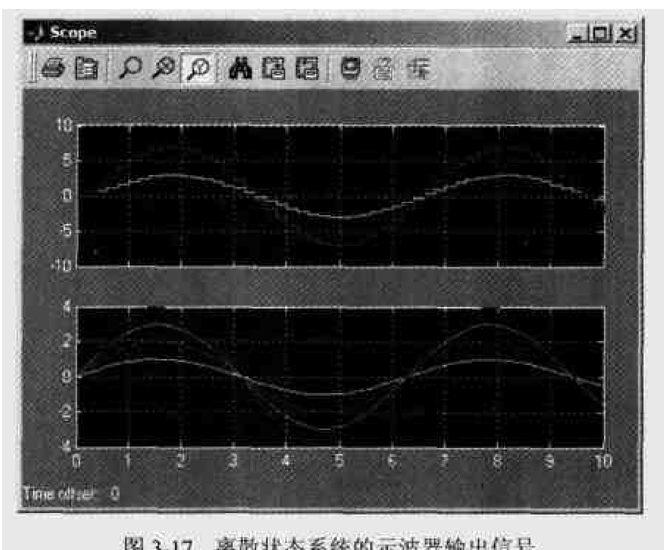



图 3-17 离散状态系统的示波器输出信号

在这个仿真模型中，S-function（S-函数模块）是一个 C 语言 S-函数，其名称为 sfun_dynsize，并且不带有任何参数。这个 C 语言 S-函数存放在 MATLAB 安装目录下的\simulink\source 中，其功能是把输入信号向量中的两个元素相加之后合并成一个信号。下面我们来看看 sfun_dynsize.c 文件的代码。

```
/* sfun_dynsize.c  输出信号向量的长度是输入信号向量长度的一半                          */
/*                 当输入端口没有信号时输出信号向量长度等于 1，输入信号向量长度等于 2   */
/*                 输出信号向量元素的数值等于输入信号向量中两个元素之和                 */
```

```
/* 定义 S-函数的名称为 sfun_dynsize */
#define S_FUNCTION_NAME   sfun_dynsize
/* S-函数的版本，必须设置为 2 */
#define S_FUNCTION_LEVEL 2
/* 文件 simstruc.h 定义了 SimStruct 结构及相关的宏 */
#include "simstruc.h"

/* Function: mdlInitializeSizes ===============================================*/
/* S-函数的初始化                                                              */
static void mdlInitializeSizes(SimStruct *S)
{
    /* 设置 S-函数参数的个数为 0 */
    ssSetNumSFcnParams(S, 0);
    /* 检查 S-函数参数数目是否与设置的数值一致 */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* 如果不一致则直接返回 */
        return;
    }

    /* 设置 S-函数连续状态的数目为 0 */
    ssSetNumContStates(S, 0);
    /* 设置 S-函数离散状态的数目为 1 */
    ssSetNumDiscStates(S, 1);

    /* 设置 S-函数输入端口的个数为 1 */
    if (!ssSetNumInputPorts(S, 1)) return;
    /* 设置 S-函数不存在直接反馈 */
    ssSetInputPortDirectFeedThrough(S, 0, 0);

    /* 设置 S-函数输出端口的个数为 1 */
    if (!ssSetNumOutputPorts(S,1)) return;

    /* 当输入端口没有连接时发出警告 */
    /* 同时把输入端口的宽度设置为 2，输出端口的宽度设置为 1 */
    if (!ssGetInputPortConnected(S,0)) {
        ssWarning(S,"input is unconnected or grounded, setting input width to 2");
        ssSetInputPortWidth(S, 0, 2);
        ssSetOutputPortWidth(S, 0, 1);
    /* 否则，输入端口和输出端口的宽度都是动态设置 */
    } else {
```

```
        ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
        ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
    }

    /* 当输出端口没有连接时发出警告 */
    if (!ssGetOutputPortConnected(S,0)) {
        ssWarning(S,"output is unconnected or terminated");
    }

    /* 设置抽样时间的个数为 1（抽样时间在 mdlInitializeSampleTimes 中进行定义） */
    ssSetNumSampleTimes(S, 1);

    /* 设置实数向量工作区的长度为动态确定 */
    ssSetNumRWork(S, DYNAMICALLY_SIZED);
    /* 设置整数向量工作区的长度为 0 */
    ssSetNumIWork(S, 0);
    /* 设置指针向量工作区的长度为 0 */
    ssSetNumPWork(S, 0);
    /* 设置工作模式向量的长度为 0 */
    ssSetNumModes(S, 0);
    /* 设置过零点检测状态的个数为 0 */
    ssSetNumNonsampledZCs( S, 0);

    /* 设置 S-函数工作模式的选项为没有异常处理 */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* 如果本函数编译成 MEX 文件则定义 mdlSetInputPortWidth 和 mdlSetOutputPortWidth 函数 */
#if defined(MATLAB_MEX_FILE)
# define MDL_SET_INPUT_PORT_WIDTH
  /* Function: mdlSetInputPortWidth =====================================*/
  /* 设置输入端口的宽度                                                    */
  static void mdlSetInputPortWidth(SimStruct *S, int_T port, int_T inputPortWidth)
  {
      /* 如果输入信号向量的长度不等于 2 的倍数则报错返回 */
      if (inputPortWidth % 2 != 0) {
          ssSetErrorStatus(S,"Input width must be a multiple of 2");
          return;
      }
      /* 否则，将输入端口的宽度设置为当前输入信号向量的长度 */
```

```
        ssSetInputPortWidth(S,port,inputPortWidth);
        /* 同时把输出端口的宽度设置为输入端口宽度的一半 */
        ssSetOutputPortWidth(S,port,inputPortWidth/2);
    }

# define MDL_SET_OUTPUT_PORT_WIDTH
  /* Function: mdlSetOutputPortWidth ==========================================*/
  /* 设置输出端口的宽度                                                              */
  static void mdlSetOutputPortWidth(SimStruct *S, int_T port, int_T outputPortWidth)
  {
        /* 根据当前输出信号向量的长度设置输入信号端口和输出信号端口的宽度 */
        /* 将输入端口的宽度设置为当前输出信号向量长度的两倍 */
        ssSetInputPortWidth(S,port,2*outputPortWidth);
        /* 将输出端口的宽度设置为当前输出信号向量的长度 */
        ssSetOutputPortWidth(S,port,outputPortWidth);
  }
#endif

/* Function: mdlInitializeSampleTimes ==========================================*/
/* 设置抽样时间                                                                      */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* 本模块的抽样时间设置为继承输入信号的抽样时间 */
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    /* 设置抽样时间偏移为固定的积分步长 */
    ssSetOffsetTime(S, 0, FIXED_IN_MINOR_STEP_OFFSET);
}

/* 如果本函数编译成 MEX 文件则定义 mdlSetWorkWidths 函数 */
#if defined(MATLAB_MEX_FILE)
# define MDL_SET_WORK_WIDTHS
  /* Function: mdlSetWorkWidths ==========================================*/
  /* 设置工作区向量的长度                                                      */
  static void mdlSetWorkWidths(SimStruct *S)
  {
      /* 实数工作区向量的长度等于输出端口的当前宽度 */
      ssSetNumRWork(S, ssGetOutputPortWidth(S,0));
  }
#endif /* defined(MATLAB_MEX_FILE) */
```

```
/* Function: mdlOutputs ====================================================*/
/* 计算输出信号的数值 y = rwork                                             */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T   i;
    /* 获得输出端口的输出信号向量 */
    real_T *y       = ssGetOutputPortRealSignal(S,0);
    /* 获得输出端口的宽度 */
    int_T   ny      = ssGetOutputPortWidth(S,0);
    /* 获得实数工作区向量 */
    real_T *rwork = ssGetRWork(S);
    /* 未使用仿真时刻参数 tid */
    UNUSED_ARG(tid);

    /* 输出信号向量每个元素的数值等于实数工作区向量的数值 */
    for (i = 0; i < ny; i++) {
        *y++ = *rwork++;
    }
}

#define MDL_UPDATE
/* Function: mdlUpdate =====================================================*/
/* 每个抽样时刻到来时更新离散状态                                           */
static void mdlUpdate(SimStruct *S, int_T tid)
{
    int_T                i;
    /* 获得输入端口的输入信号指针向量 */
    InputRealPtrsType uPtrs   = ssGetInputPortRealSignalPtrs(S,0);
    /* 获得输出端口的宽度 */
    int_T                ny      = ssGetOutputPortWidth(S,0);
    /* 获得实数工作区向量 */
    real_T               *rwork = ssGetRWork(S);
    /* 未使用仿真时刻参数 tid */
    UNUSED_ARG(tid); /* not used in single tasking mode */

    /* 实数工作区向量每个元素的数值等于输出信号向量中两个相邻元素的数值之和 */
    for (i = 0; i < ny; i++) {
        *rwork++ = *uPtrs[2*i] + *uPtrs[2*i+1];
    }
}
```

```
/* Function: mdlTerminate ============================================*/
/* 执行仿真结束时的清理工作                                             */
static void mdlTerminate(SimStruct *S)
{
    /* 未使用仿真结构参数 S */
    UNUSED_ARG(S);
}

/* 如果本函数编译成 MEX 文件则链接 simulink.c 文件  */
/* 否则，链接 cg_sfun.h 文件                        */
#ifdef   MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

最后，在 MATLAB 工作区中输入命令行“mex sfun_dynsize.c”，得到一个 MEX 文件 sfun_dynsize.dll。这个动态链接库文件在仿真开始的时候被 Simulink 调入内存，Simulink 将根据仿真进程的需要调用相应的函数获取仿真结果。

## 3.4 C++语言 S-函数

C++语言实际上是在 C 语言中引入了类和对象的概念，因此，C++语言 S-函数的代码结构在很大程度上与 C 语言 S-函数相似。为了保留 Simulink 仿真过程中的 C 语言调用方式，C++语言 S-函数除了类定义代码采用 C++语言规范之外，其他与 Simulink 交互的代码都采用 C 语言进行编写。为此，在编译 C++语言 S-函数时，需要通过“extern ‘C’”宏把 C++编译器设置为 C 语言风格。

为了使读者更好地理解 C++语言 S-函数的基本结构，我们通过一个简单的 C++语言 S-函数来介绍 C++语言 S-函数的代码编写过程。在这个示例程序中，C++类 counter 从 0 开始计数，并且通过 output()函数输出这个计数值。这个示例程序取自 MATLAB 安装目录的 /simulink/src/下的文件 sfun_counter_cpp.cpp。下面我们介绍这个 S-函数的代码。

```
/*   File     : sfun_counter_cpp.cpp                                  */
/*            输出信号向量的长度是输入信号向量长度的一半                   */
/*            当输入端口没有信号时输出信号向量长度等于 1，输入信号向量长度等于 2  */
/*            输出信号向量元素的数值等于输入信号向量中两个元素之和           */

#include "iostream.h"
/* 定义一个 C++类 counter */
class   counter {
    /* 定义类中的私有变量 x */
```

```
    double   x;
public:
    /* 定义类的构造函数 */
    counter() {
        x = 0.0;
    }
    /* 定义类的输出信号 */
    double output(void) {
        x = x + 1.0;
        return x;
    }
};

/* 通知C++编译器采用C调用方式对下面的代码进行编译 */
#ifdef __cplusplus
extern "C" {
#endif

/* S-函数的版本，必须设置为2 */
#define S_FUNCTION_LEVEL 2
/* 定义S-函数的名称为sfun_counter_cpp */
#define S_FUNCTION_NAME   sfun_counter_cpp
/* 文件simstruc.h定义了SimStruct结构及相关的宏 */
#include "simstruc.h"

/* Function: mdlInitializeSizes ==============================================*/
/* S-函数的初始化                                                              */
static void mdlInitializeSizes(SimStruct *S)
{
    /* 设置S-函数参数的个数为1（该参数表示抽样时间） */
    ssSetNumSFcnParams(S, 1);
    /* 检查S-函数参数数目是否与设置的数值一致 */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* 如果不一致则直接返回 */
        return;
    }

    /* 设置S-函数连续状态的数目为0 */
    ssSetNumContStates(S, 0);
    /* 设置S-函数离散状态的数目为0 */
```

```
    ssSetNumDiscStates(S, 0);

    /* 设置 S-函数输入端口的个数为 0 */
    if (!ssSetNumInputPorts(S, 0)) return;
    /* 设置 S-函数输出端口的个数为 1 */
    if (!ssSetNumOutputPorts(S, 1)) return;
    /* 设置 S-函数输出端口的宽度为 1 */
    ssSetOutputPortWidth(S, 0, 1);
    /* 设置抽样时间的个数为 1（抽样时间在 mdlInitializeSampleTimes 中进行定义） */
    ssSetNumSampleTimes(S, 1);
    /* 设置实数向量工作区的长度为 0 */
    ssSetNumRWork(S, 0);
    /* 设置整数向量工作区的长度为 0 */
    ssSetNumIWork(S, 0);
    /* 设置指针向量工作区的长度为 1 */
    ssSetNumPWork(S, 1);
    /* 设置工作模式向量的长度为 0 */
    ssSetNumModes(S, 0);
    /* 设置过零点检测状态的个数为 0 */
    ssSetNumNonsampledZCs(S, 0);
    /* 设置 S-函数工作模式的选项 */
    ssSetOptions(S, 0);
}

/* Function: mdlInitializeSampleTimes ========================================*/
/* 设置抽样时间                                                              */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* 本模块的抽样时间由输入参数确定 */
    ssSetSampleTime(S, 0, mxGetScalar(ssGetSFcnParam(S, 0)));
    /* 设置抽样时间偏移为 0 */
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_START
#if defined(MDL_START)
  /* Function: mdlStart ==============================================*/
  /* 仿真开始时的操作                                                  */
  static void mdlStart(SimStruct *S)
  {
```

```
    /* 仿真开始时创建一个 counter 对象，其指针保存在指针工作区向量中 */
    ssGetPWork(S)[0] = (void *) new counter;
  }
#endif /*  MDL_START */

/* Function: mdlOutputs =================================================*/
/* 计算输出信号的数值 y = rwork                                          */
static void mdlOutputs(SimStruct *S, int_T tid)
{
   /* 从指针工作区中获得 counter 对象 */
   counter *c = (counter *) ssGetPWork(S)[0];
   /* 获得输出端口的输出信号向量 */
   real_T   *y = ssGetOutputPortRealSignal(S,0);
   /* 输出信号由 counter 对象中的 output 方法确定 */
   y[0] = c->output();
}

/* Function: mdlTerminate ===============================================*/
/* 执行仿真结束时的清理工作                                              */
static void mdlTerminate(SimStruct *S)
{
   /* 从指针工作区中获得 counter 对象 */
   counter *c = (counter *) ssGetPWork(S)[0];
   /* 仿真结束，删除 counter 对象 */
   delete c;
}

/* 如果本函数编译成 MEX 文件则链接 simulink.c 文件 */
/* 否则，链接 cg_sfun.h 文件                        */
#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif

/* extern "C"宏的结尾 */
#ifdef __cplusplus
}
#endif
```

最后，在 MATLAB 工作区中输入命令行参数“mex sfun_counter_cpp.cpp”对 C++语言

S-函数进行编译。需要注意的是，由于 C++语言 S-函数中使用了 iostream.h 文件，因此，在编译之前需要设置该文件所在的目录，或者是采用 Microsoft 的 Visual C/C++编译器进行编译。下面的代码是在 MATLAB 工作区中设置编译器的过程（中文部分是添加的注释）。

```
>> mex -setup
Please choose your compiler for building external interface (MEX) files:
## 输入回车或按键 y
Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Digital Visual Fortran version 6.0 in C:\Program Files\Microsoft Visual Studio
[2] Lcc C version 2.4 in C:\MATLAB6.5\sys\lcc
[3] Microsoft Visual C/C++ version 7.0 in C:\Program Files\Microsoft Visual Studio .NET
[4] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft Visual Studio
[0] None
## 输入 4，选择 Microsoft Visual C/C++ version 6.0 编译器
Compiler: 4

Please verify your choices:
## 检查 Microsoft Visual C/C++ version 6.0 编译器的安装路径是否正确
Compiler: Microsoft Visual C/C++ 6.0
Location: C:\Program Files\Microsoft Visual Studio

## 输入 y 确认编译器的安装路径
Are these correct?([y]/n): y

The default options file:
"C:\Documents and Settings\deng\Application Data\MathWorks\MATLAB\R13\mexopts.bat"
is being updated from C:\MATLAB6.5\BIN\WIN32\mexopts\msvc60opts.bat...

Installing the MATLAB Visual Studio add-in ...
  Updated C:\Program Files\Microsoft Visual Studio\common\msdev98\template\MATLABWizard.awx
    from C:\MATLAB6.5\BIN\WIN32\MATLABWizard.awx
  Updated C:\Program Files\Microsoft Visual Studio\common\msdev98\template\MATLABWizard.hlp
    from C:\MATLAB6.5\BIN\WIN32\MATLABWizard.hlp
  Updated C:\Program Files\Microsoft Visual Studio\common\msdev98\addins\MATLABAddin.dll
    from C:\MATLAB6.5\BIN\WIN32\MATLABAddin.dll
  Merged C:\MATLAB6.5\BIN\WIN32\usertype.dat
    with C:\Program Files\Microsoft Visual Studio\common\msdev98\bin\usertype.dat

Note: If you want to use the MATLAB Visual Studio add-in with the MATLAB C/C++
```

```
Compiler, you must start MATLAB and run the following commands:
cd(prefdir);
mccsavepath;
(You only have to do this configuration step once.)

## 开始编译 sfun_counter_cpp.cpp 文件
>> mex sfun_counter_cpp.cpp
```

编译 sfun_counter_cpp.cpp 之后，得到动态链接库文件 sfun_counter_cpp.dll。为了验证这个 C++语言 S-函数的功能，我们设计了一个名为 sfundemo_counter_cpp.mdl 的仿真模型，其系统组成如图 3-18 所示。

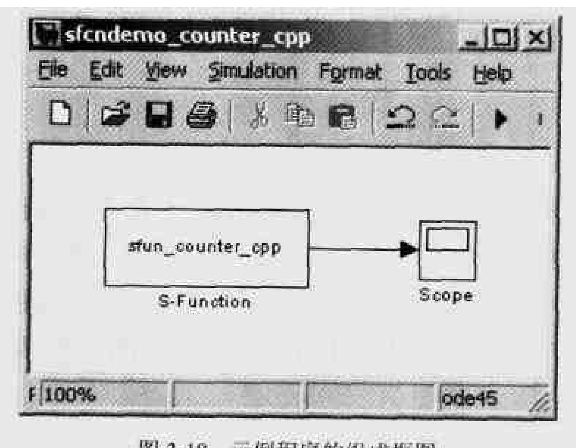



图 3-18 示例程序的组成框图

在这个仿真模型中，S-Function（S-函数模块）中的参数 S-function name 设置为 sfun_counter_cpp，参数 S-function parameters 可以设置为任意一个正数，它表示仿真的抽样间隔。S-函数模块的输出信号连接到 Scope（示波器模块），图 3-19 所示分别是 S-函数模块的参数 S-function parameters 设置为 0.5 和 1 时示波器的输出波形。

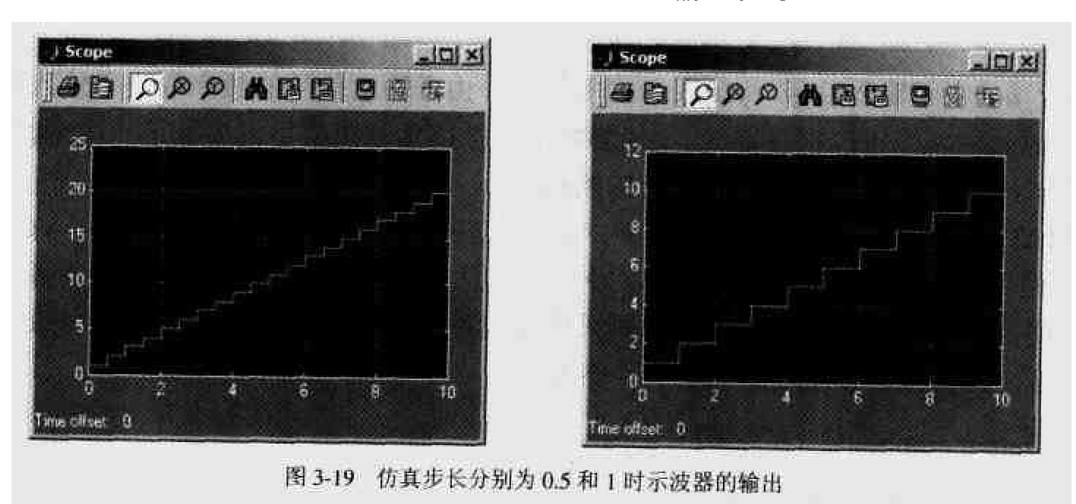



图 3-19 仿真步长分别为 0.5 和 1 时示波器的输出

从图 3-19 所示中可以看出，S-函数模块的抽样间隔取决于该模块的参数，并且这个 C++语言 S-函数 sfun_counter_cpp 在每个抽样周期内的输出信号依次递增，这就是我们在 counter 类的 output()函数中实现的功能。

# 第 4 章　信源和信宿

通信系统一般由 3 个部分组成，即信源、信道和信宿。信源是通信系统的起点，它产生数据并且对这些数据进行编码和调制，产生适合于信道传输的调制信号；信宿则是通信系统的终点，它从信道中接收信号，通过解码和解调得到信源端产生的原始数据。

从广义上说，信源和信宿除了完成数据的产生和接收外，还包括信号的编码和译码，以及信号的调制和解调。本章先介绍数据的产生和接收，即狭义的信源和信宿。

## 4.1　信源

本节所说的信源是指产生信息的装置，它表达了发送端需要传送的内容。信源产生的信息即可以是模拟信号，也可以是数字信号。我们每天使用的固定电话的时候，通话双方产生的语音就是一个模拟信号，它的幅度是连续变化的。现在的通信系统大多采用数字调制方式，这时候信源产生的就是数字信号，或者是经过量化编码的模拟信号。

### 4.1.1　压控振荡器

压控振荡器 VCO（Voltage-Controlled Oscillator）是指输出信号的频率随着输入信号幅度的变化而发生相应的变化的设备，它的工作原理可以通过公式 4.1 来描述。

$$y(t)=A_c\cos\left(2\pi f_c t+2\pi k_c\int_0^t u(\tau)d\tau+\varphi\right) \tag{4.1}$$

其中，$u(t)$表示输入信号，$y(t)$表示输出信号。由于输出信号的频率取决于输入信号电压的变化，因此称为“压控振荡器”。其他影响压控振荡器输出信号的参数还有信号幅度 $A_c$，振荡频率$f_c$，输入信号灵敏度 $k_c$，以及初始相位 $\varphi$。

在 MATLAB 中，压控振荡器有两种：离散时间压控振荡器（Discrete-Time VCO）和连续时间压控振荡器（Continuous-Time VCO）。这两种压控振荡器的差别在于，前者对输入信号 $u(t)$采用离散方式进行积分，而后者则采用连续积分，它的参数设置框图分别如图 4-1 和图 4-2 所示。压控振荡器主要有以下几个参数。

■　Output amplitude（输出信号幅度）

压控振荡器输出信号的幅度。

■　Oscillation frequency（振荡频率）

振荡频率表示输入信号为 0 时输出信号的频率（单位：Hz）。

■　Input sensitivity（输入信号灵敏度）

输入信号灵敏度对应于公式 4.1 中的参数 $k_c$，它对输入信号的幅度进行缩放，通过这种

方式影响输出信号的频率。输入信号灵敏度的物理意义是：当输入信号的强度增加或降低 1 V 时，输出信号的频率变化的幅度。

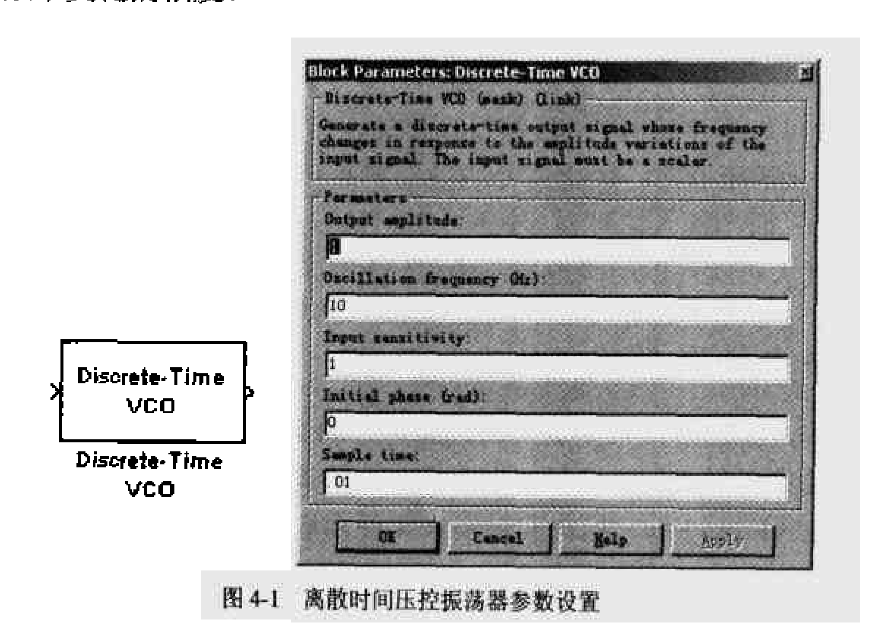



图 4-1　离散时间压控振荡器参数设置

■　**Initial phase（初始相位）**

初始相位对应于公式 4.1 中的参数 $\varphi$，表示振荡器产生的输出信号的初始相位（单位：弧度）。

■　**Sample time（抽样时间）**

本参数只在离散时间压控振荡器中有效，它表示离散积分的抽样间隔。在连续时间压控振荡器中不需要制定积分的抽样间隔。

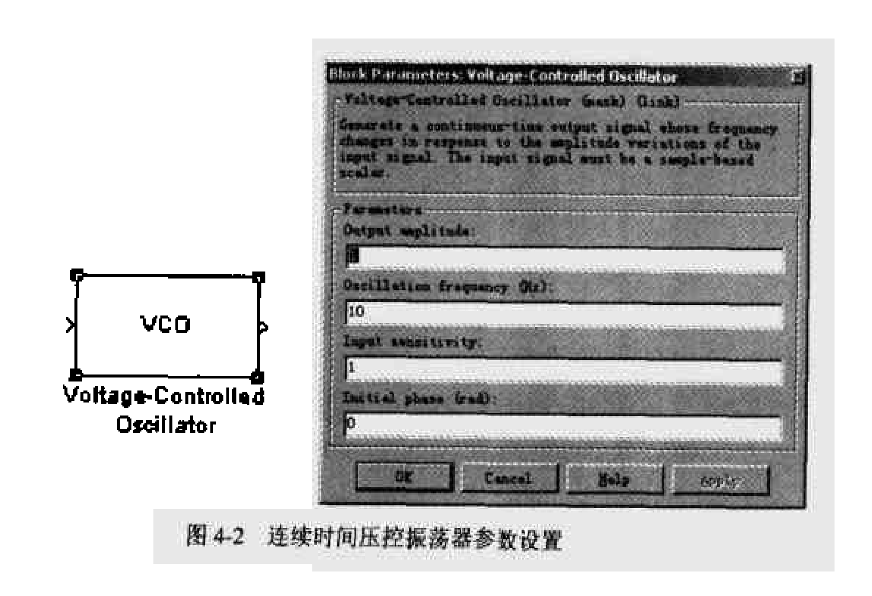



图 4-2　连续时间压控振荡器参数设置

为了更清楚地考察压控振荡器输出信号的频率与输入信号幅度之间的关系，我们对公式 4.1 作出变换，取输出信号的角度 $\Delta$，

$$\Delta = 2\pi f_c t + 2\pi\, k_c \int_0^t u(\tau)d\tau + \varphi \tag{4.2}$$

对输出信号的角度 $\Delta$ 求积分，得到输出信号的角频率 $\omega$ 和频率 $f$，

$$\omega = 2\pi f_c + 2\pi k_c u(t) \tag{4.3}$$

$$f = \frac{\omega}{2\pi} = f_c + k_c u(t) \tag{4.4}$$

从公式 4.4 中可以清楚地看到，压控振荡器输出信号的频率 $f$ 与输入信号的幅度 $u(t)$ 成正比。当输入信号 $u(t)$ 等于 0 时，输出信号的频率 $f$ 等于 $f_c$；当输入信号 $u(t)$ 大于 0 时，输出信号的频率 $f$ 高于 $f_c$；而当输入信号 $u(t)$ 小于 0 时，输出信号的频率 $f$ 就低于 $f_c$。这样，通过改变输入信号的幅度大小就可以准确地控制输出信号的频率。

## 4.1.2 从文件中读取数据

在仿真过程中，我们常常希望从一个已经写好的文件中读入事先准备好的数据，或者是把某次仿真产生的结果保存到文件里，作为另一个仿真的输入。MATLAB 为此提供了两种仿真模块：文件读取（From File）模块和触发式文件读取（Triggered Read From File）模块。这两个模块都能够从文件中提取数据，它们之间的差别在于，前者一次性地从文件中读取所有的数据并且把它们保存到工作区里，而后者是“触发式”的，它在输入信号的上升沿的驱动下从文件中读取下一个数据。

### 1. 文件读取模块

文件读取模块（From File）从指定路径的指定文件中读取数据，如图 4-5 所示。文件读取模块的文件路径和名字由参数 File name 确定，并且这个文件必须组织成一个（$m, n$）矩阵的形式，如公式 4.5 所示。这个（$m, n$）矩阵的行数必须不小于 2，即 $m \ge 2$。矩阵的第一行 $t_j(1 \le j \le n)$ 是单调增加的时间点。从第二行开始，矩阵的每个元素 $u_j^i$ 是与它所在的列 $t_j$ 对应的时间点的数值。

$$\begin{pmatrix} t_1 & t_2 & \cdots & t_n \\ u_1^1 & u_2^1 & \cdots & u_n^1 \\ \cdots & \cdots & \cdots & \cdots \\ u_1^{m-1} & u_2^{m-1} & \cdots & u_n^{m-1} \end{pmatrix} \tag{4.5}$$

文件读取模块的输出信号的维数取决于文件中的矩阵的行数（等于 $m-1$），并且根据矩阵第一行的时间点输出与该时间点相对应的一列数据（但是不包括第一行）。关于文件读取模块的另外两个特性是关于它的内插（Interpolate）和外推（Extrapolate）。

对于处于矩阵中两个时间点 $t_i$ 和 $t_{i+1}$ 中的任意时刻 $t$，MATLAB 采用内插的方式计算该时刻的输出，根据公式 4.6，由 $u_i^k$ 和 $u_{i+1}^k$ 的数值计算 $t$ 时刻第 $k$ 个输出的数值 $u_t^k$。

$$u_t^k = u_i^k + \frac{t - t_i}{t_{i+1} - t_i} \times \left(u_{i+1}^k - u_i^k\right), t_{i+1} \ne t_i \tag{4.6}$$

当 $t_i = t_{i+1}$ 时，矩阵的第一行存在着两个相同的时间点，MATLAB 选择序号较小的列对应的数值作为数值或内插运算的端点。

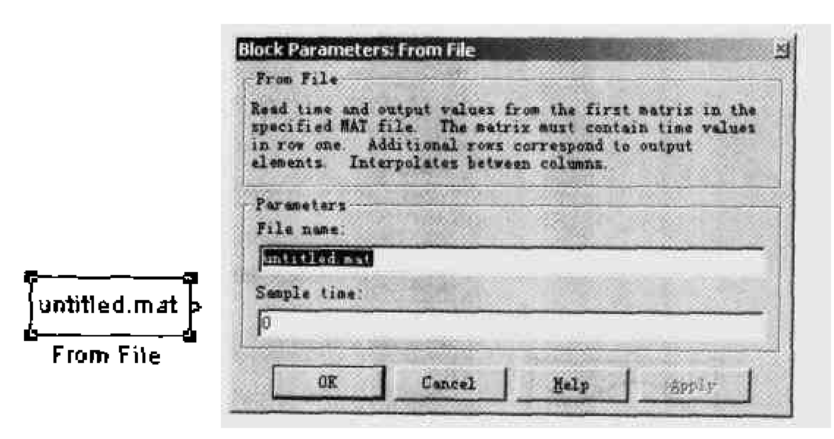



图 4-3　文件读取模块及其参数设置

在早于时刻$t_1$或晚于时刻$t_n$的时刻$t$，MATLAB 根据最近的两个时间点的数值采用外推的方法计算时刻$t$的数值，该时刻第 $k$ 个输出的数值由公式 4.7 确定。

$$\begin{cases} u_t^k = u_1^k + \dfrac{t-t_1}{t_2-t_1}\times\left(u_2^k - u_1^k\right), t_2 \neq t_1, \text{当} t < t_1 \text{时} \\ u_t^k = u_{n-1}^k + \dfrac{t-t_{n-1}}{t_n-t_{n-1}}\times\left(u_n^k - u_{n-1}^k\right), t_n \neq t_{n-1}, \text{当} t > t_n \text{时} \end{cases} \tag{4.7}$$

在文件读取模块中，除了用于指定文件路径和名称的参数 File name 之外，还有另外一个参数 Sample time，即抽样间隔。抽样间隔是文件读取模块进行内插运算和外推运算的周期，MATLAB 根据这个抽样间隔，要么从文件的矩阵中读取该时刻对应的数值，要么通过内插或外推计算出输出信号。当 Sample time 等于 0 时表示对信号实施连续抽样。

文件读取模块主要有以下两个参数。

■　File name（文件名称）

文件的完整名称，包括文件的扩展名及其所在的路径。

■　Sample time（抽样时间）

文件读取模块的抽样间隔。

应该注意的是，不能在同一个模型中同时使用文件读取模块（From File）和文件写入模块（To File）对同一个文件进行操作。文件写入模块将在本章的 4.2.3 节进行介绍。

### 2. 触发式文件读取模块

触发式文件读取模块（Triggered Read From File）也是从指定的文件中读取数据，但是它并不是通过文件中指定的时间点来获取当前时刻的数据，而是通过一个输入信号的上升沿作为触发信号，触发 MATLAB 从指定的文件中读取下一行数据。触发式文件读取模块及其参数设置如图 4-4 所示。

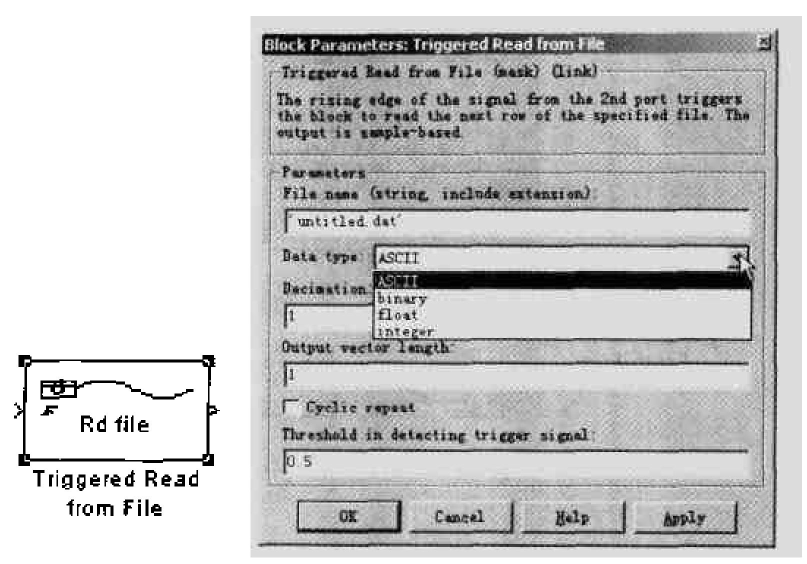



图 4-4 触发式文件读取模块及其参数设置

触发式文件读取模块的参数主要有以下几个。

■ File name（文件名称）

文件的完整名称，包括文件的扩展名及其所在的路径。

■ Data type（数据类型）

触发式文件读取模块可以读取 4 种类型的数据，即字符型（ASCII）、二进制型（Binary）、浮点型（Float）以及整型（Integer）。当触发式文件读取模块读入字符型（ASCII）数据时，MATLAB 根据 ASCII 字符表把每一个读入的字符转化成相应的整数作为输出，如空格转换成 32，字符 A 转换成 65，而字符 a 则转换成 97。在仿真过程中，通常需要通过触发式文件写入模块（Triggered Write To File）把数据保存到文件中，然后通过触发式文件读取模块读入这些数据，这时候就要求两种模块中的数据类型保持一致。

■ Decimation（抽取率）

抽取率等于 1 时，触发式文件读取模块在每个输入信号的上升沿都执行一次读取操作；当抽取率大于 1 时，触发式文件读取模块每隔若干个上升沿读取一次文件，这个间隔由抽取率（Decimation）指定。不管抽取率是否大于 1，触发式文件读取模块在输入信号的第一个上升沿都会读一次文件。

■ Output vector length（输出向量长度）

触发式文件读取模块每次从文件中读取固定长度的数据，这个长度由输出向量长度（Output vector length）确定。值得注意的是，这个长度是跟文件的数据类型有关的，它指的是特定类型的数据的个数。

■ Cyclic repeat（循环读取）

如果选中了循环读取（Cyclic repeat）复选框，当触发时文件读取模块到达文件末尾时，它将自动回到文件的开始位置重新读取数据；否则，触发时文件读取模块输出 0。

■ Threshold in detecting trigger signal（触发门限）

触发信号的门限。当输入信号的强度由低至高达到触发门限时，MATLAB 认为出现了一个上升沿。

### 3. 从工作区中读取数据

除了前面介绍的两种从文件中读取数据的模块之外，另外一个比较重要的输入模块是工作区读取模块（From Workspace），用于从工作区中读取数据。工作区读取模块及其参数设置如图 4-5 所示。与文件读取模块和触发式文件读取模块不同的是，工作区读取模块从工作区中读取某个变量的数值。在工作区读取模块到达变量的末尾时，它既可以采用外推的方式计算下一个输出，也可以把此后的输出设置为 0 或最后一个数值，或者采用循环读取的方式。工作区读取模块的操作与文件读取模块由很多相似之处。

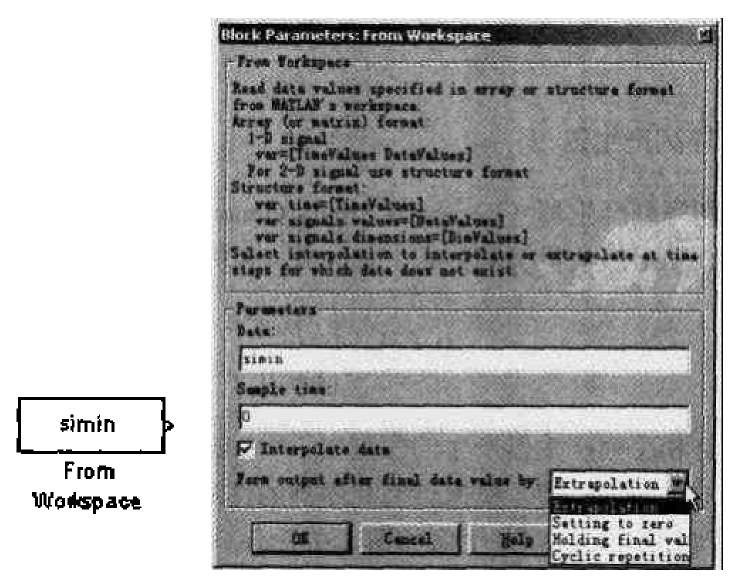



图 4-5　工作区读取模块及其参数设置

工作区读取模块主要有以下几个参数。

■　Data（输入数据）

包含仿真时间和与这些时间点对应的数值的向量。假设向量 T 是一个关于仿真时间的向量，向量 U 是与向量 T 重的每一个元素相对应的数据，则 Data 就等于[T U]。

■　Sample time（抽样时间）

工作区读取模块的抽样间隔。

■　Interpolate data（内插数据）

当选择了 Interpolate data 选项时，工作区读取模块用内插的方法计算输入数据中未指定的时间点的数值；否则，工作区读取模块将选取与最近的一个时间点对应的数据作为这个时间点的数值。

■　Form output after final data value by（末尾数据的处理方式）

在工作区读取模块读取工作区数据的过程中，如果当前的仿真时间大于工作区数据的最后一个指定的时刻，MATLAB 提供了 4 种不同的方式来计算这些时刻的数值：外推方式（Extrapolation）、置零方式（Setting To Zero）、保持方式（Holding Final Value）以及循环读取方式（Cyclic Repetition）。

### 4.1.3 数据源

在一个通信系统中，信源产生的信号有两种：模拟信号和数字信号。关于模拟信号的一个例子就是我们日常打电话时产生的语音信号。在公共交换电话网络 PSTN 以及第一代模拟蜂窝移动网络中，发送端直接对语音信号进行编码和调制。这种方式存在着诸多的问题，因此，从第二代蜂窝移动网络开始，发送端首先对模拟的语音信号进行量化和编码，产生一个数字序列，然后对这个数字序列进行处理。另外，现在的各种核心网络都只传输数字信号，数字信号已经成为现代通信系统不可或缺的信号源。本节将介绍几种数字信号产生器，包括二进制贝努利序列产生器、二进制误差样本产生器、泊松整数产生器以及随机整数产生器。

**1. 二进制贝努利序列产生器**

二进制贝努利序列产生器（Bernoulli Binary Generator）产生一个二进制序列，并且这个二进制序列中的 0 和 1 服从贝努利分布，如公式 4.8 所示。

$$\Pr(x)=\begin{cases} p, x=0 \\ 1-p, x=1 \end{cases} \tag{4.8}$$

即二进制贝努利序列产生器产生的序列中，1 出现的概率为 $p$，0 出现的概率为 $1-p$，其中 $p$ 是介于 0 和 1 之间的实数。根据贝努利序列的性质可知，该序列的均值为 $1-p$，方差等于 $p(1-p)$。二进制贝努利序列产生器如图 4-6 所示。

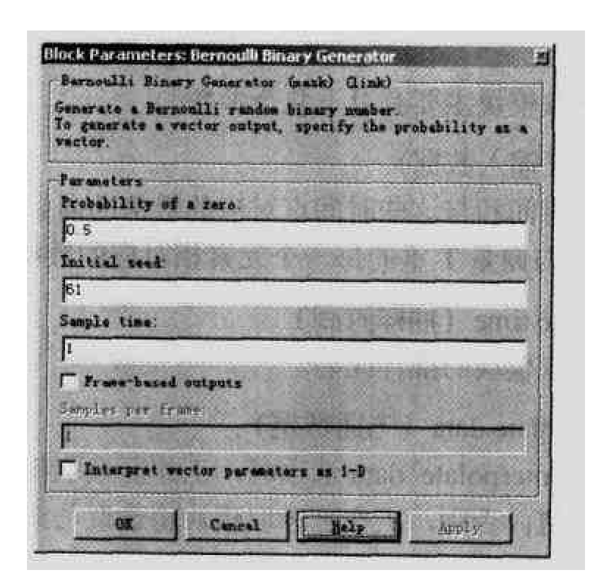



图 4-6 二进制贝努利序列产生器模块及其参数设置

二进制贝努利序列产生器主要有以下几个参数。

- **Probability of a zero（出现 0 的概率）**

二进制贝努利序列产生器产生的二进制序列中 0 出现的概率，对应于公式 4.8 中的参数 $p$，该数值必须是介于 0 和 1 之间的某个实数。

■ Initial seed（随机数种子）

二进制贝努利序列产生器的随机数种子。当使用相同的随机数种子时，二进制贝努利序列产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，二进制贝努利序列产生器的输出信号的维数也大于 1。例如，设置随机数种子为[1 2 5]，则输出的序列是一个三维向量。

■ Sample time（抽样时间）

输出序列中每个二进制符号的持续时间。

■ Frame-based outputs（帧格式输出）

指定二进制贝努利序列产生器以帧格式产生输出序列。如果选择了该选项，就不能再选择参数“Interpret vector parameters as 1-D”。

■ Samples per frame（每帧的抽样数）

当选择了“Frame-based outputs”参数之后，本参数用来确定每帧的抽样点的数目。

■ Interpret vector parameters as 1-D（产生一维向量）

如果选择了该选项，二进制贝努利序列产生器产生一维的输出序列，这时候不能选择“Frame-based outputs”复选框；否则，输出序列是一个二维向量。

图 4-7 所示是在 $p = 0.5$，Sample time 等于 0.1 秒时的一个二进制贝努利序列的波形。从图 4-7 所示中可以看到，当 $p = 0.5$ 时，序列中 0 和 1 出现的概率基本上是相等的。

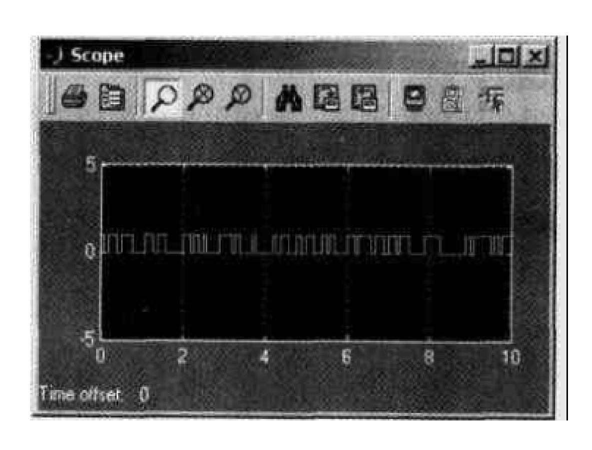



图 4-7 一个二进制贝努利序列（$p$=0.5）

## 2. 二进制误差样本产生器

二进制误差样本产生器（Binary Error Pattern Generator）输出一个二进制序列，并且这个序列中根据某种该率出现 1（即误差）。二进制误差样本产生器与二进制贝努利序列产生器的主要差别在于，二进制误差样本产生器能够指定一系列概率[$p_1, p_2, \ldots p_n$]，使得在输出序列中出现单个 1 的概率为 $p_1$，连续出现两个 1 的概率为 $p_2$，以此类推，连续出现 $n$ 个 1 的概率为 $p_n$。二进制误差样本产生器经常用于仿真错误控制编码的性能。应该注意的是这些概率的和应该小于 1，而且 $n$ 应该小于指定的输出序列的长度。二进制误差样本产生器如图 4-8 所示。

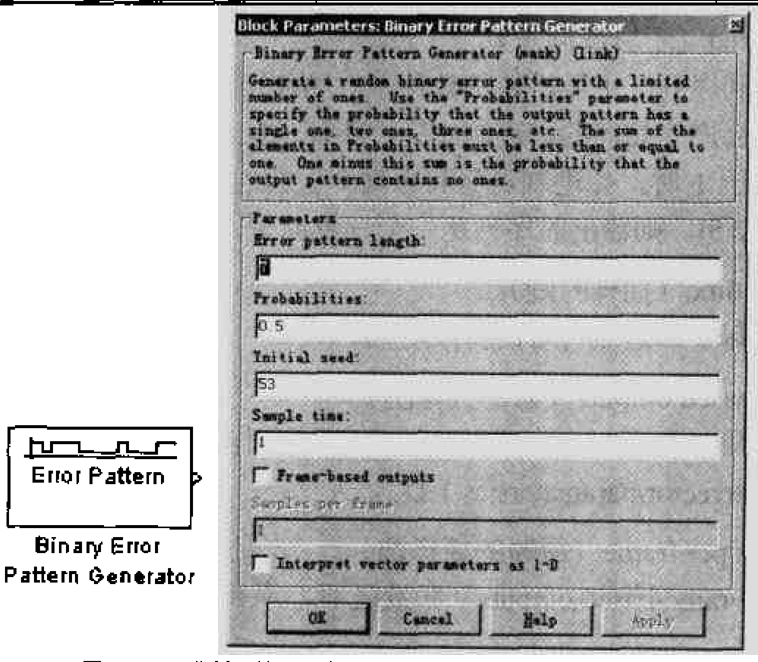



图 4-8　二进制误差样本产生器模块及其参数设置

二进制误差样本产生器主要有以下几个参数。

■　Error pattern length（误差样本长度）

二进制误差样本产生器产生的输出序列的长度。

■　Probabilities（误差概率）

误差概率$[p_1, p_2, \ldots p_n]$指定了出现连续 $i$ 个 1 的概率 $p_i$。

■　Initial seed（随机数种子）

二进制误差样本产生器的随机数种子。当使用相同的随机数种子时，二进制误差样本产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。

■　Sample time（抽样时间）

输出序列中每个二进制符号的持续时间。

■　Frame-based outputs（帧格式输出）

指定二进制误差样本产生器以帧格式产生输出序列。如果选择了该选项，就不能再选择参数 Interpret vector parameters as 1-D。

■　Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■　Interpret vector parameters as 1-D（产生一维向量）

如果选择了该选项，二进制误差样本产生器产生一维的输出序列，这时候不能选择 Frame-based outputs 选项；否则，输出序列是一个二维向量。

另外一个产生二进制误差样本的方法是使用命令行方式的函数 randerr()，它产生一个指定长度的矩阵，矩阵中每一行的元素相互独立。函数 randerr()有以下几种形式：

■　out = randerr(m)

产生一个 m 行 m 列的二进制序列，每一行有且只有一个元素不等于 0。

■　out = randerr(m,n)

产生一个 m 行 n 列的二进制序列，每一行有且只有一个元素不等于 0。

■ out = randerr(m,n,errors)

产生一个 $m$ 行 $n$ 列的二进制序列，每一行非零元素的个数由参数 errors 确定。当 errors 是一个标量时，矩阵的每一行都有相同个数的非零元素，非零元素的个数等于 errors；当 errors 是一个行向量时，它指定了输出矩阵每一行可能出现的非零元素的个数，errors 中的每个元素出现的概率相等；当 errors 是一个两行的矩阵时，errors 的第一行指定了输出矩阵各行可能出现的非零元素的最大个数，errors 的第二行的元素 errors(2, $k$)则表示输出矩阵每一行出现 errors(1, $k$)个非零元素的概率。

下面的程序段说明了 randerr 函数的用法。

```
>> randerr(8,7,[1 3])
ans =
     0     0     0     0     0     1     0
     0     0     0     1     0     0     0
     0     0     1     1     0     0     1
     0     1     0     1     0     1     0
     0     0     0     0     0     1     0
     1     0     0     1     1     0     0
     1     1     0     1     0     0     0
     0     0     0     1     0     0     0
>> randerr(8,7,[1 3; 0.25 0.75])
ans =
     0     0     0     0     0     1     0
     0     0     1     0     1     0     1
     0     1     0     0     1     1     0
     0     0     0     1     0     0     0
     0     0     1     1     1     0     0
     1     1     1     0     0     0     0
     1     0     0     0     0     1     1
     0     1     1     0     1     0     0
```

■ out = randerr(m,n,errors,state)

本函数首先把随机数种子设置为 state，然后执行与 out = randerr(m,n,errors)相同的操作。

### 3. 泊松整数产生器

泊松整数产生器（Poisson Integer Generator）产生一个服从泊松分布的整数序列。设 $X$ 是一个服从泊松分布的随机变量，则 $X$ 可以取任意一个自然数，且 $X$ 等于自然数 $k$ 的概率由公式 4.9 确定。

$$\Pr(k)=\frac{\lambda^k e^{-\lambda}}{k!},k=0,1,2\cdots \tag{4.9}$$

在公式 4.9 中，λ 是泊松随机过程的参数，并且泊松随机过程的均值和方差都等于 λ。泊松整数产生器如图 4-9 所示。

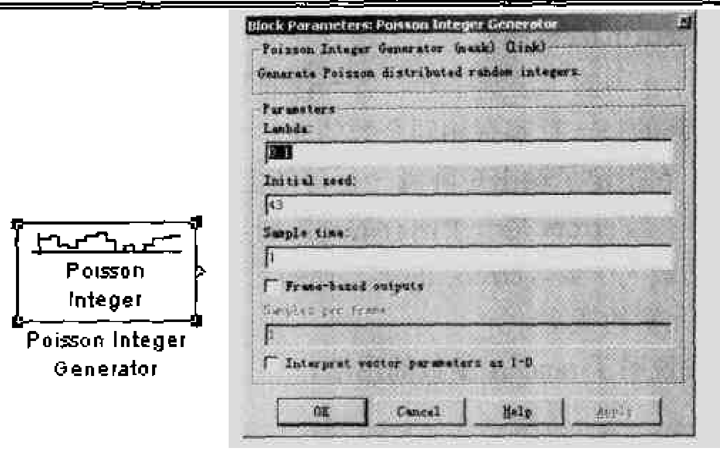



图 4-9　泊松整数产生器模块及其参数设置

通信系统建模中通常把呼叫到达间隔和服务间隔假定为服从泊松分布，在这种情况下，参数 λ 一般都小于 1。泊松整数产生器主要有以下几个参数。

■　Lambda（泊松参数 λ）

用于确定泊松随机过程的参数 λ。

■　Initial seed（随机数种子）

泊松整数产生器的随机数种子。当使用相同的随机数种子时，泊松整数产生器每次都会产生相同的整数序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，泊松整数产生器的输出信号的维数也大于 1。

■　Sample time（抽样时间）

输出序列中每个整数的持续时间。

■　Frame-based outputs（帧格式输出）

指定泊松整数产生器以帧格式产生输出序列。如果选择了该选项，就不能再选择参数 Interpret vector parameters as 1-D。

■　Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■　Interpret vector parameters as 1-D（产生一维向量）

如果选择了该选项，泊松整数产生器产生一维的输出序列，这时候不能选择 Frame-based outputs 选项；否则，输出序列是一个二维向量。

图 4-10 所示是当参数 λ 分别等于 0.1 和 10 时泊松整数产生器的输出波形。从图 4-10 所示中可以看到，当 λ 等于 0.1 时（见左图所示），输出序列的最大值等于 3，而当 λ 等于 10 时（见右图所示），输出序列的平均值大致等于 10。

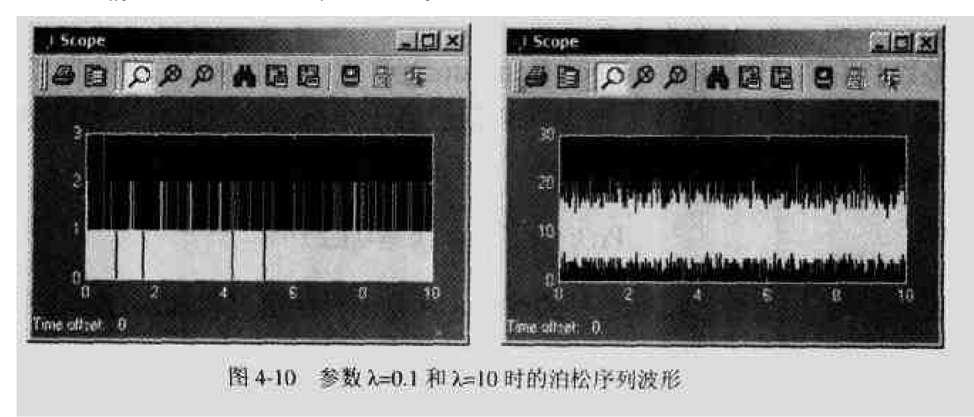
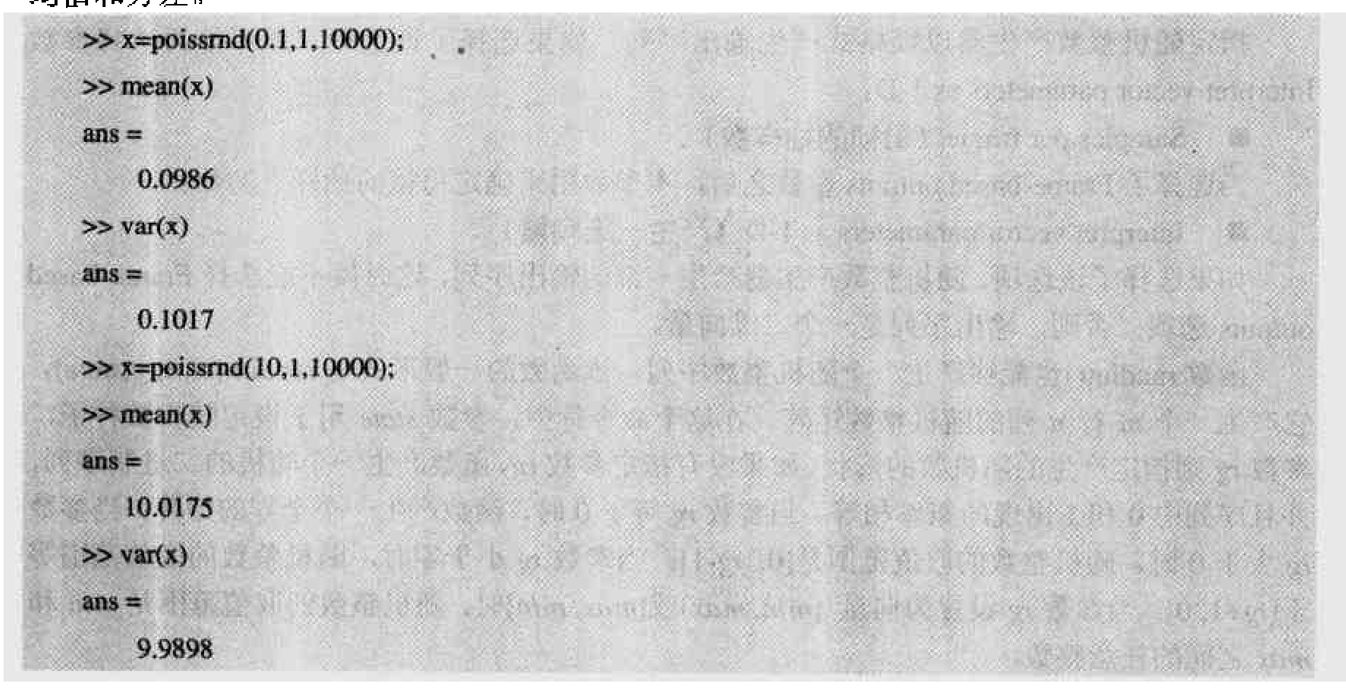



图 4-10　参数 λ=0.1 和 λ=10 时的泊松序列波形

MATLAB 还提供了一个函数 poissrnd()，用来产生一个泊松随机整数序列。下面的程序段分别产生 λ 等于 0.1 和 10 时具有 10000 个元素的泊松整数序列，并且验证了这两个序列的均值和方差。



```
>> x=poissrnd(0.1,1,10000);
>> mean(x)
ans =
    0.0986
>> var(x)
ans =
    0.1017
>> x=poissrnd(10,1,10000);
>> mean(x)
ans =
   10.0175
>> var(x)
ans =
    9.9898
```

## 4. 随机整数产生器

随机整数产生器（Random Integer Generator）用来产生一个在 0 和 $M-1$ 之间均匀分布的随机整数序列。随机整数产生器模块及其参数设置对话框如图 4-11 所示。

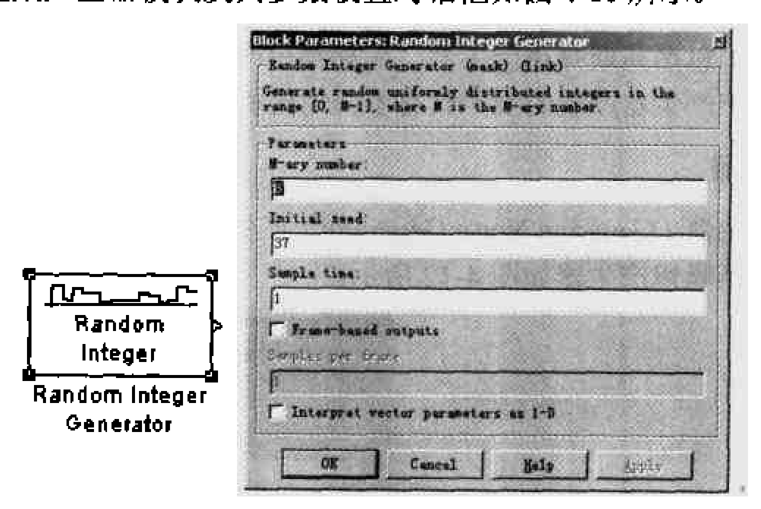



图 4-11　随机整数产生器模块及其参数设置

随机整数产生器主要有以下几个参数。

■　M-ary number（输出范围）

设定随机整数的取值范围。当该参数设置为 $M$ 时，随机整数的取值范围等于[0, $M$-1]。

■　Initial seed（随机数种子）

随机整数产生器的随机数种子。当使用相同的随机数种子时，随机整数产生器每次都会产生相同的整数序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，随机整数产生器的输出信号的维数也大于 1。

■ Sample time（抽样时间）

输出序列中每个整数的持续时间。

■ Frame-based outputs（帧格式输出）

指定随机整数产生器以帧格式产生输出序列。如果选择了该选项，就不能再选择参数 Interpret vector parameters as 1-D。

■ Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■ Interpret vector parameters as 1-D（产生一维向量）

如果选择了该选项，随机整数产生器产生一维的输出序列，这时候不能选择 Frame-based outputs 选项；否则，输出序列是一个二维向量。

函数 randint()也能够产生一个随机整数序列。该函数的一般形式是 randint(*m*,*n*,*rg*,*state*)，它产生一个 *m* 行 *n* 列的随机整数矩阵。在这个命令行中，参数 *state* 用于设定随机数种子，参数 *rg* 则指定产生的随机数的形状。如果没有指定参数 *rg*，函数产生一个随机的二进制序列，并且序列中 0 和 1 出现的概率相等。当参数 *rg* 等于 0 时，函数产生一个全零的矩阵；当参数 *rg* 大于 0 时，随机整数的取值范围是[0, *rg*-1]；当参数 *rg* 小于零时，随机整数的取值范围等于[*rg*+1, 0]；当参数 *rg* 设置为向量 [*min*, *max*]或[*max*, *min*]时，随机整数的取值范围是 *min* 和 *max* 之间的任意整数。

## 4.1.4 噪声源

在通信系统中，噪声是无处不在的，它在很大程度上影响着信号传输的质量。噪声本身是一个随机过程，很难通过一种简单的计算方式预测某个时刻噪声信号的强度。本节将介绍几种噪声产生器，包括高斯噪声产生器、瑞利噪声产生器、伦琴噪声产生器以及均匀噪声产生器，这几种噪声产生器产生的噪声具有不同的随机特征。

### 1. 高斯噪声产生器

高斯噪声产生器（Gaussian Noise Generator）产生离散的噪声信号，并且这种噪声信号服从高斯分布。高斯噪声产生器如图 4-12 所示。

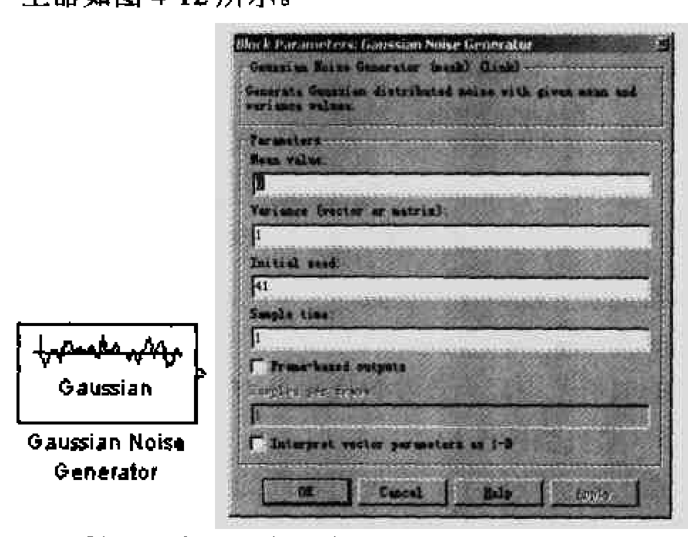



图 4-12 高斯噪声产生器模块及其参数设置

对于一维的高斯随机变量 $X$，如果它的均值为 $\mu$，方差等于 $\sigma^2$，则随机变量 $X$ 取值为 $x$

的概率 $p(x)$ 由公式 4.10 确定。

$$p(x)=\frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{(2\pi)}\sigma} \tag{4.10}$$

图 4-13 所示是当方差 $\sigma^2$ 分别等于 0.5、1 和 2 时一维高斯随机过程的概率密度。

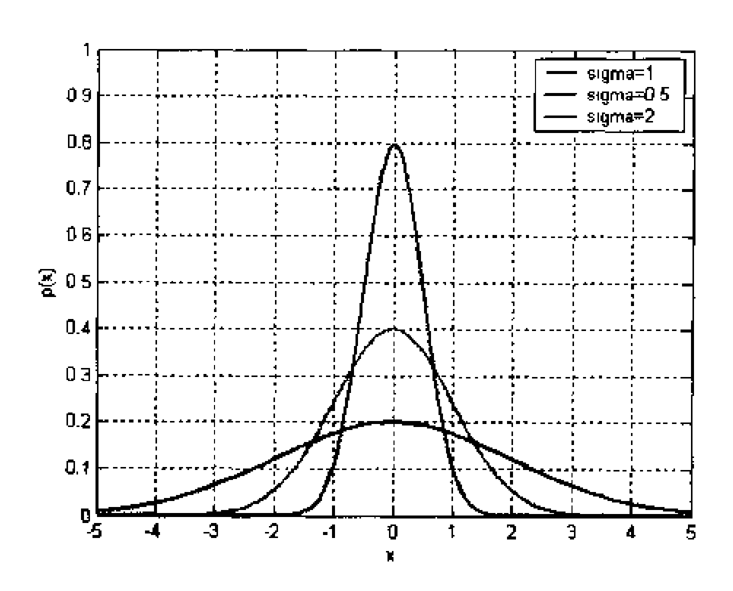



图 4-13　一维高斯随机过程的概率密度

对于 n 维随机变量 $X=(X_1, X_2,\ldots, X_n)$，如果它的均值为 $\mu=(\mu_1, \mu_2, \ldots, \mu_n)$，协方差矩阵为 $K$，则 $X$ 等于 $x$ 的概率 $p(x)$ 由公式 4.11 确定。

$$p(x)=\frac{e^{-\frac{(x-\mu)^T K^{-1}(x-\mu)}{2}}}{\sqrt{(2\pi)^n \det K}} \tag{4.11}$$

在公式 4.11 中，$T$ 表示矩阵的转置运算，$\det K$ 表示取协方差矩阵 $K$ 的行列式的值。

高斯噪声产生器主要有以下几个参数。

■　Mean value（均值）

高斯随机变量 $X$ 的均值 $\mu$。当 $\mu$ 是一个标量时，高斯噪声产生器产生的噪声服从一维高斯分布；当 $\mu$ 是一个 $n$ 维向量时，输出的噪声服从 $n$ 维高斯分布。

■　Variance（方差）

本参数设置高斯随机变量 $X$ 的方差 $\sigma^2$ 或协方差矩阵 $K$。当 Variance 是一个标量时，高斯噪声产生器产生的噪声服从一维高斯分布，方差 $\sigma^2$ 就等于 Variance；当 Variance 是一个 $n$ 维向量时，输出的噪声服从 $n$ 维高斯分布，协方差矩阵 $K$ 的对角线元素等于 Variance，非对角线元素都等于零，这时候输出向量的任意两个元素之间是不相关的；当 Variance 是一个 $n$ 行 $n$ 列的矩阵时，协方差矩阵 $K$ 等于 Variance。

需要注意的一点是，对于一维高斯变量，方差应该是一个正数；对于多维高斯变量，协方差矩阵应该是一个正定矩阵。

■　Initial seed（随机数种子）

高斯噪声产生器的随机数种子。当使用相同的随机数种子时，高斯噪声产生器每次都会产生相同的整数序列；不同的随机数种子通常产生不同的序列。当随机数种子是一个向量，

且它的维数等于 $n$ 时，高斯噪声产生器的输出信号服从 $n$ 维高斯分布。

■ Sample time（抽样时间）

输出序列中每个整数的持续时间。

■ Frame-based outputs（帧格式输出）

指定高斯噪声产生器以帧格式产生输出序列。如果选择了该选项，就不能再选择参数 Interpret vector parameters as 1-D。

■ Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■ Interpret vector parameters as 1-D（产生一维向量）

如果选择了该选项，高斯噪声产生器产生一维的输出序列，这时候不能选择 Frame-based outputs 选项；否则，输出序列是一个二维向量。

图 4-14 所示是当均值为 0，方差为 1，抽样周期为 0.05 秒时高斯噪声产生器的一个输出波形。

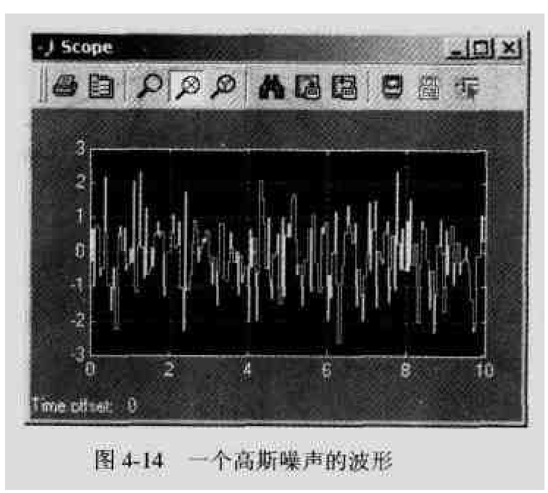



图 4-14 一个高斯噪声的波形

### 2. 瑞利噪声产生器

瑞利噪声产生器（Rayleigh Noise Generator）产生一个服从瑞利分布的随机噪声信号，其参数设置框图如图 4-15 所示。

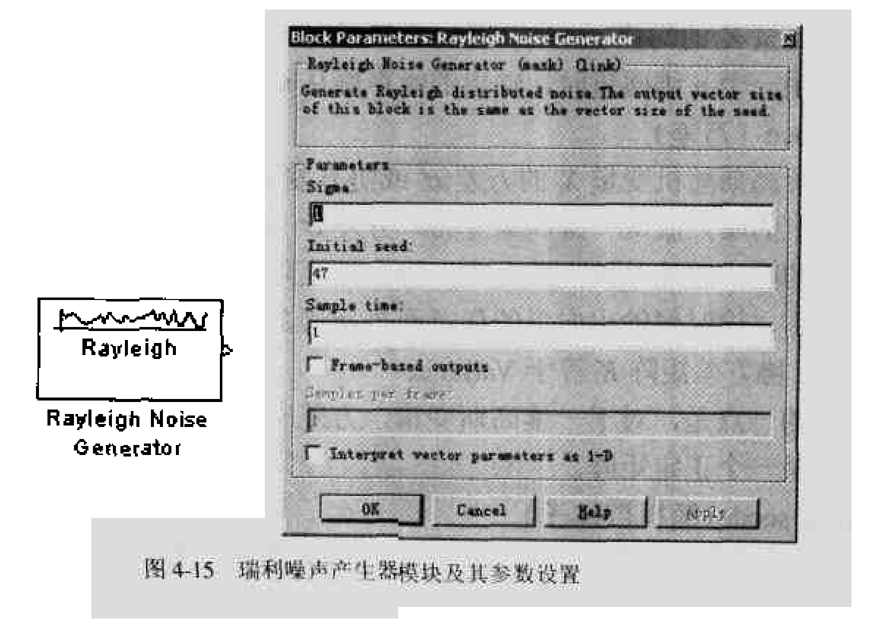



图 4-15 瑞利噪声产生器模块及其参数设置

根据瑞利分布的定义，如果一个随机变量 $X$ 服从瑞利分布，则它的概率密度函数 $p(x)$ 由公式 4.12 确定。

$$p(x)=\begin{cases}\dfrac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}},x\geq 0\\ 0,x<0\end{cases} \tag{4.12}$$

公式 4.12 中的 $\sigma^2$ 是决定瑞利分布的参数，称为瑞利分布的衰减包络（Fading Envelope）。瑞利随机变量 $X$ 的均值等于 $\sqrt{\pi/2}\sigma$，方差等于 $(4-\pi)\sigma^2/2$。图 4-16 所示为 σ 分别等于 0.5、1 和 2 时瑞利随机变量的概率分布曲线。

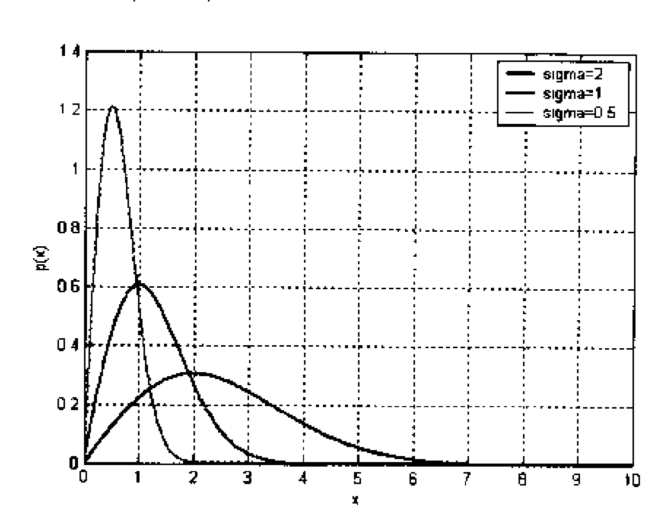



图 4-16　瑞利随机过程的概率密度

瑞利噪声产生器主要有以下几个参数。

- Sigma（衰减包络）

用于确定瑞利随机过程的参数，对应于公式 4.12 中的参数 $\sigma$。

- Initial seed（随机数种子）

瑞利噪声产生器的随机数种子。当使用相同的随机数种子时，瑞利噪声产生器每次都会产生相同的整数序列；不同的随机数种子通常产生不同的序列。当随机数种子是一个向量，且它的维数等于 $n$ 时，瑞利噪声产生器的输出是一个 $n$ 维向量。一般情况下随机数种子应该取大于 30 的素数。

- Sample time（抽样时间）

输出序列中每个整数的持续时间。

- Frame-based outputs（帧格式输出）

指定瑞利噪声产生器以帧格式产生输出序列。如果选择了该选项，就不能再选择参数 Interpret vector parameters as 1-D。

- Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■ Interpret vector parameters as 1-D（产生一维向量）

如果选择了该选项，瑞利噪声产生器产生一维的输出序列，这时候不能选择 Frame-based outputs 选项；否则，输出序列是一个二维向量。

图 4-17 所示是当参数 $\sigma$ 等于 1，抽样周期设置为 0.05 秒时瑞利噪声产生器产生的一个输出波形。

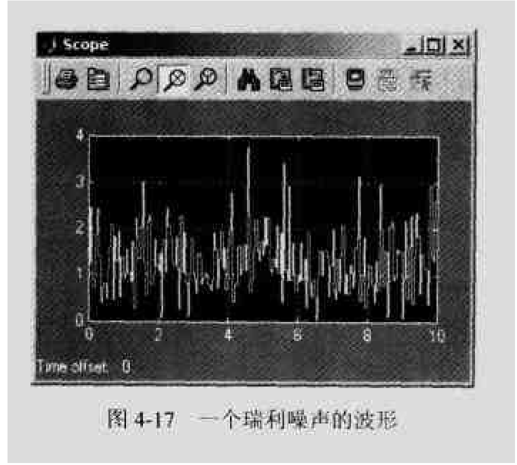



图 4-17 一个瑞利噪声的波形

### 3. 伦琴噪声产生器

伦琴噪声产生器（Rician Noise Generator）产生一个服从伦琴分布的随机噪声信号，其参数设置框图如图 4-18 所示。

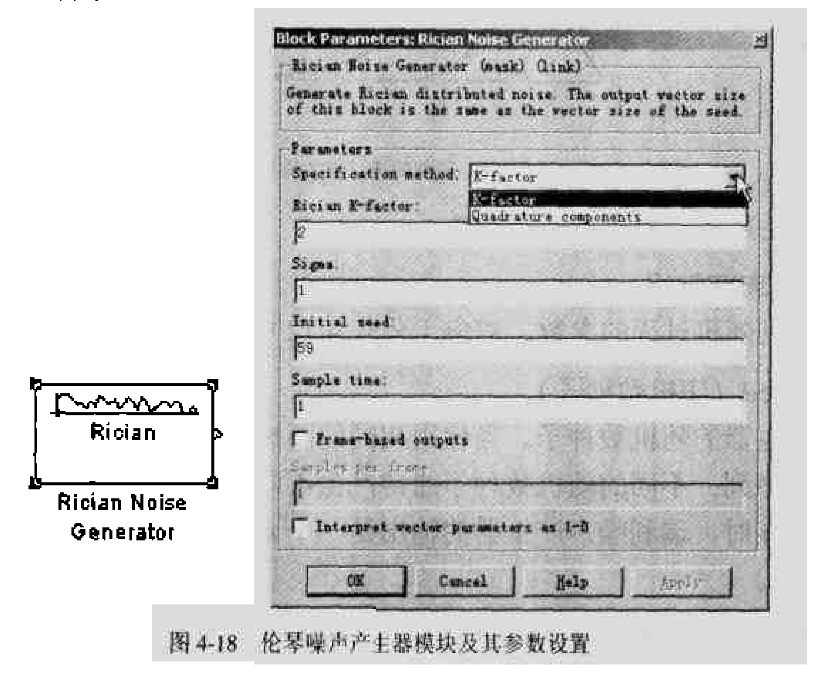



图 4-18 伦琴噪声产生器模块及其参数设置

根据伦琴分布的定义，如果一个随机变量 $X$ 服从伦琴分布，则它的概率密度函数 $p(x)$ 由公式 4.13 确定。

$$p(x)=\begin{cases}\dfrac{x}{\sigma^2}I_0\left(\dfrac{mx}{\sigma^2}\right)e^{-\frac{x^2+m^2}{2\sigma^2}},x\geq 0\\ 0,x<0\end{cases} \tag{4.13}$$

其中，$I_0(y)$是第一类零阶修正贝塞尔函数（modified 0th-order Bessel function of the first

kind)。

$$I_0(y) = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{y\cos(t)}dt \tag{4.14}$$

由于伦琴分布是根据两个独立的高斯分布构造的，假设这两个高斯分布的均值分别等于 $m_I$ 和 $m_Q$，标准方差等于 $\sigma$，则公式 4.13 中的参数 $m^2 = m_I^2 + m_Q^2$。事实上，当 $m$ 等于 0 时，伦琴分布就等于瑞利分布。

伦琴噪声产生器主要有以下几个参数。

■　Specification method（参数模式）

伦琴噪声产生器可以通过两种方式确定伦琴分布的参数：K 参数模式（K-factor）或正交分量模式（Quadrature components）。

■　K-factor（伦琴 K 参数）

当参数模式（Specification method）设置为 K-factor 时本参数有效，并且 $K = m^2/(2\sigma^2)$，其中 $m$ 和 $\sigma$ 是公式 4.13 中的参数。

■　In-phase component (mean), Quadrature component (mean)（正交分量均值）

当参数模式（Specification method）设置为 Quadrature components 时本参数有效，分别表示两个高斯分量的均值 $m_I$ 和 $m_Q$。

■　Sigma（标准方差）

对应于公式 4.13 中的参数 $\sigma$。需要注意的是此参数不是伦琴分布的标准方差。

■　Initial seed（随机数种子）

伦琴噪声产生器的随机数种子。当使用相同的随机数种子时，伦琴噪声产生器每次都会产生相同的噪声序列；不同的随机数种子通常产生不同的序列。当随机数种子是一个向量，且它的维数等于 $n$ 时，伦琴噪声产生器的输出是一个 $n$ 维向量。

■　Sample time（抽样时间）

输出序列中每个整数的持续时间。

■　Frame-based outputs（帧格式输出）

指定伦琴噪声产生器以帧格式产生输出序列。如果选择了该选项，就不能再选择参数 Interpret vector parameters as 1-D。

■　Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■　Interpret vector parameters as 1-D（产生一维向量）

如果选择了该选项，伦琴噪声产生器产生一维的输出序列，这时候不能选择 Frame-based outputs 选项；否则，输出序列是一个二维向量。

图 4-19 所示是当参数 $\sigma$ 等于 1，$K$ 等于 2，抽样周期设置为 0.05 秒时伦琴噪声产生器产生的一个输出波形。

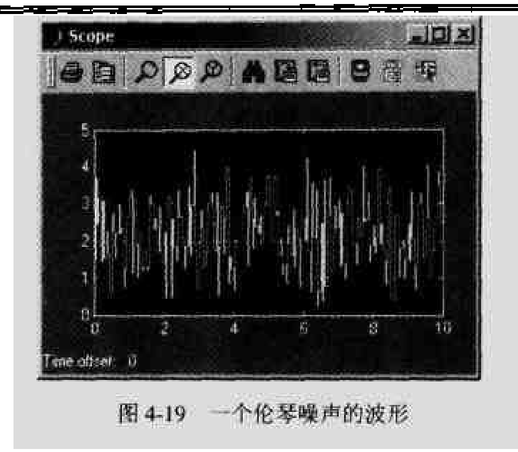
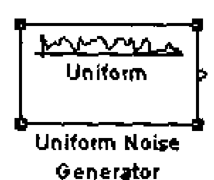


图 4-19 一个伦琴噪声的波形

### 4. 均匀噪声产生器

均匀噪声产生器（Uniform Noise Generator）产生的噪声信号在[*min*, *max*]之间均匀分布，其参数设置框图如图 4-20 所示。



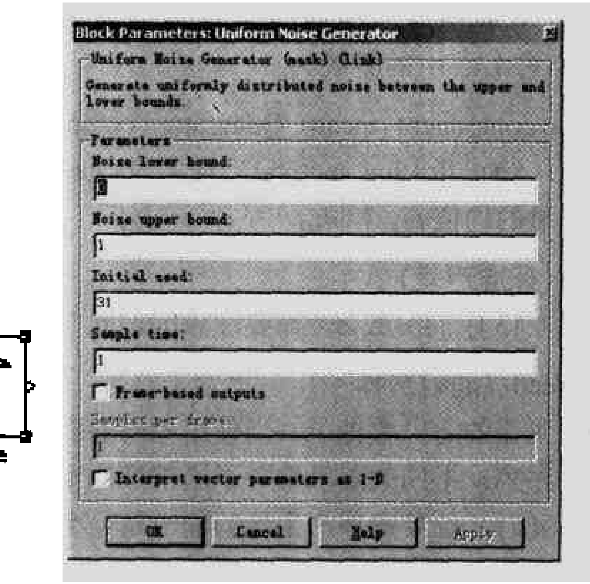



图 4-20 均匀噪声产生器模块及其参数设置

均匀噪声产生器主要有以下几个参数。

■ Noise lower bound（噪声下界）

均匀分布[*min*, *max*]的下界 *min*。

■ Noise upper bound（噪声上界）

均匀分布[*min*, *max*]的上界 *max*。

■ Initial seed（随机数种子）

均匀噪声产生器的随机数种子。当使用相同的随机数种子时，均匀噪声产生器每次都会产生相同的整数序列；不同的随机数种子通常产生不同的序列。当随机数种子是一个向量，且它的维数等于 *n* 时，均匀噪声产生器的输出信号是一个 *n* 维向量。

■ Sample time（抽样时间）

输出序列中每个元素的持续时间。

■ Frame-based outputs（帧格式输出）

指定均匀噪声产生器以帧格式产生输出序列。如果选择了该选项，就不能再选择参数 Interpret vector parameters as 1-D。

■　Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■　Interpret vector parameters as 1-D（产生一维向量）

如果选择了该选项，均匀噪声产生器产生一维的输出序列，这时候不能选择 Frame-based outputs 选项；否则，输出序列是一个二维向量。

图 4-21 所示是当均匀分布的下界为 1，上界为 3，抽样周期等于 0.05 秒时均匀噪声产生器产生的一个输出波形。

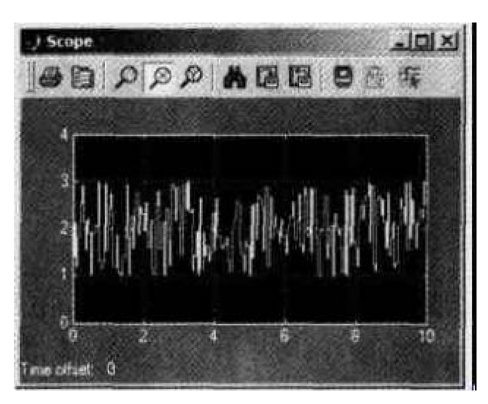



图 4-21　一个均匀噪声的波形

## 4.1.5　序列生成器

序列生成器用来产生一个具有某种特性的二进制序列，这种序列可能有比较独特的自相关属性或互相关属性。在码分多址移动通信系统中，通过使用某种序列来达到多址接入效果，如 CDMA 2000 使用了 Walsh 码和 PN 序列，WCDMA 中还使用了 Gold 序列。

假设 $X$ 是一个长度为 $n$ 的二进制序列 $X$（在计算码的相关性时通常表示成+1 和－1 的序列，其中+1 等价于二进制序列中的 0，－1 等价于二进制序列的 1），则序列 $X$ 的自相关函数 $C_k(X)$定义为 $X$ 的一个移位序列（或循环移位序列）与原序列 $X$ 的内积。自相关函数由两种计算方式：循环方式和非循环方式，公式 4.15 是这两种方法的计算公式。

$$C_k(X)=\begin{cases}\sum_{i=1}^{n}X_iX_{(i+k)\bmod n},X_i\in\{+1,-1\},\ \text{循环方式}\\ \sum_{i=1}^{n-k}X_iX_{(i+k)},X_i\in\{+1,-1\},\text{非循环方式}\end{cases} \tag{4.15}$$

类似地，对于两个长度为 $n$ 的二进制序列 $X$ 和 $Y$，序列中的元素是+1 和－1，则这两个序列的互相关函数 $R_k(X,Y)$定义为 $Y$ 的一个移位序列（或循环移位序列）与原序列 $X$ 的内积，如公式 4.16 所示。

$$R_k(X,Y)=\begin{cases}\sum_{i=1}^{n}X_iY_{(i+k)\bmod n},X_i\in\{+1,-1\},\ \ Y_i\in\{+1,-1\},\text{循环方式}\\ \sum_{i=1}^{n-k}X_iY_{(i+k)},X_i\in\{+1,-1\},Y_i\in\{+1,-1\},\text{非循环方式}\end{cases} \tag{4.16}$$

MATLAB 提供了若干种序列生成器，在本节中我们将依次介绍 Hadamard 码生成器、Walsh 码生成器、PN 序列生成器、Barker 码生成器、Gold 序列生成器、OVSF 码生成器以及 Kasami 序列生成器。

### 1. Hadamard 码生成器

Hadamard 码生成器（Hadamard Code Generator）输出 Hadamard 矩阵中的一个 Hadamard 码序列。Hadamard 矩阵中的每一个行向量都是一个 Hadamard 码序列，每个行向量都与其他行向量保持正交。这种正交性对于扩频通信系统来说是至关重要的，只有保持正交性才能够从多个接收信号中识别出所需的信号来。

Hadamard 矩阵 $H_N$ 是一个 $N$ 行 $N$ 列的矩阵，矩阵中的元素是+1 或－1，并且 $N$ 满足条件 $N=2^n, n=0,1,2\cdots$。Hadamard 矩阵 $H_N$ 可以通过递归的方式构造，即

$$\begin{cases} H_1=[+1] \\ H_{2N}=\begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix} \end{cases} \tag{4.17}$$

关于 Hadamard 矩阵 $H_N$ 的一个重要性质是 $H_N H_N^T = N I_N$，其中 $I_N$ 是 $N$ 阶单位****矩阵。

另外，对于同一个 Hadamard 矩阵 $H_N$ 中的任意两个 Hadamard 码序列 $X$ 和 $Y$，它们的互相关函数都等于零，即对于任意的 $k$，$R_k(X,Y)$等于 0。CDMA 系统就是利用这种特性来区分不同的信道。Hadamard 码生成器如图 4-22 所示。

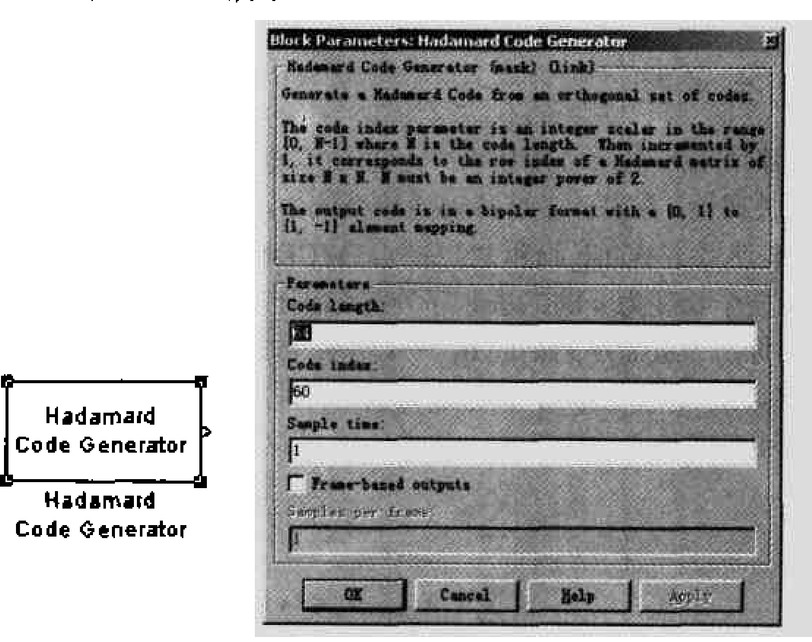



图 4-22　Hadamard 码生成器模块及其参数设置

Hadamard 码生成器主要有以下几个参数。

■　Code length（码长）

Hadamard 码的长度，对应于 Hadamard 矩阵的参数 $N$。

■　Code index（码序号）

用于确定 Hadamard 码生成器输出哪一个的序列。如果 Hadamard 码的长度等于 $N$，则 Code index 是介于 0 和 $N-1$ 之间的一个整数，MATLAB 取 Hadamard 矩阵中的第 Code index 行作为输出。

■　Sample time（抽样时间）

输出序列中每个元素的持续时间。

■　Frame-based outputs（帧格式输出）

指定 Hadamard 码生成器以帧格式产生输出序列。

■　Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

2. Walsh 码生成器

Walsh 码生成器（Walsh Code Generator）产生一个 Walsh 码。长度为 $N$ 的 Walsh 码实际上是一个 $N$ 阶 Hadamard 矩阵的一个行向量，因此 Walsh 码也具有上一节中介绍的 Hadamard 码的性质。Walsh 码生成器如图 4-23 所示。

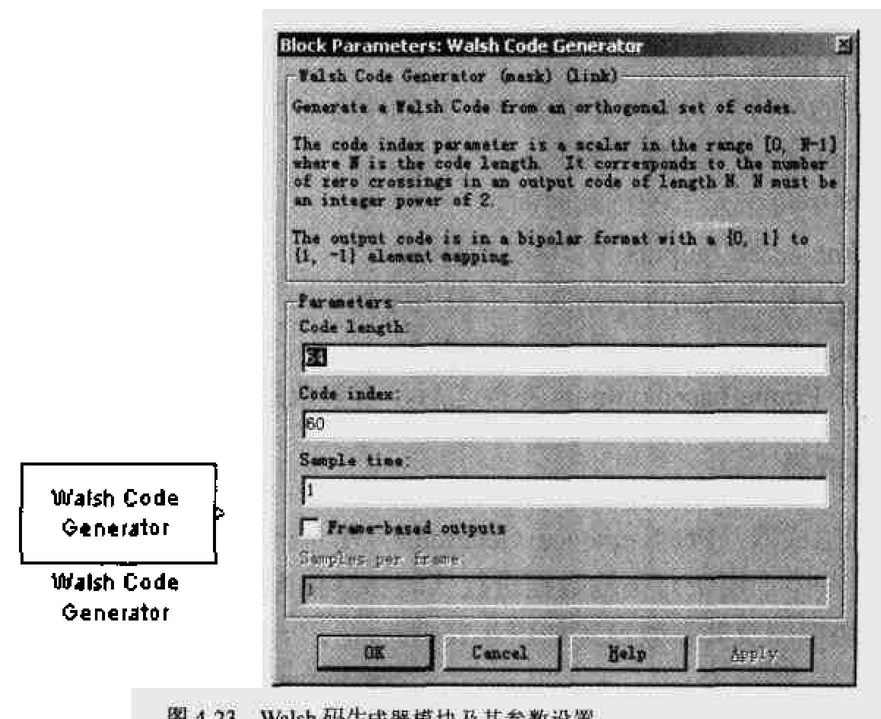



图 4-23　Walsh 码生成器模块及其参数设置

如果用 $W_i$ 表示第 $i$ 个长度为 $N$ 的 Walsh 码，其中 $i$ 是介于 0 和 $N-1$ 之间的整数，且 Walsh 码的元素是+1 或−1，$W_i[k]$ 表示 Walsh 码序列 $W_i$ 的第 $k$ 个元素，则对于任意的 $i$，$W_i[0]=0$。对于任意两个长度为 $N$ 的 Walsh 码 $W_i$ 和 $W_j$，有：

$$W_i W_j^T = \begin{cases} 0, i \neq j \\ N, i = j \end{cases}$$

另外需要注意的一点是，Walsh 码生成器通过码序列中过零点的数目来标识相同长度的各个码序列，这跟 Hadamard 码生成器的码序号是不同的。对于长度为 $N$ 的 Walsh 码 $W_i$，它恰好有 $i$ 个过零点。图 4-24 所示是长度为 64，码序号分别等于 17 和 63 的两个 Walsh 码序列（抽样周期等于 10/64 秒），它们分别有 17 和 63 个过零点。

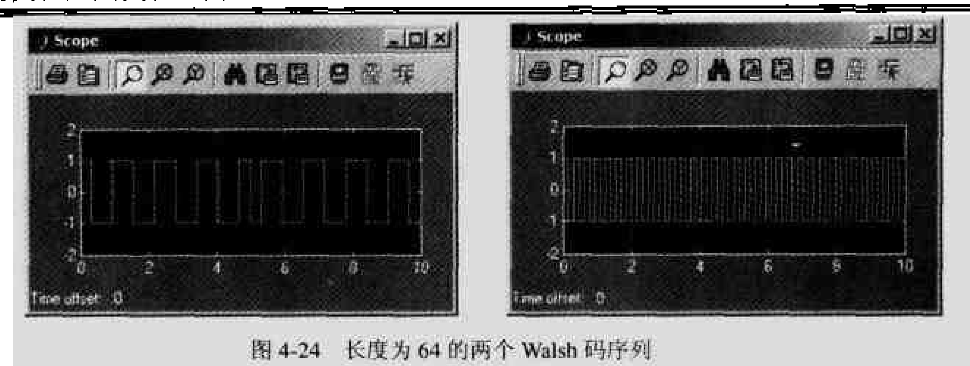



图 4-24　长度为 64 的两个 Walsh 码序列

Walsh 码生成器主要有以下几个参数。

■　Code length（码的长度）

Walsh 码的长度 $N$，且满足条件 $N=2^n, n=0,1,2\cdots$。

■　Code index（码序号）

Walsh 码的序号，它是介于 0 和 $N-1$ 之间的一个整数，标是序列中过零点的数目。

■　Sample time（抽样时间）

输出序列中每个元素的持续时间。

■　Frame-based outputs（帧格式输出）

指定 Walsh 码生成器以帧格式产生输出序列。

■　Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

### 3. PN 序列生成器

PN 序列生成器（PN Sequence Generator）用于产生一个伪随机序列。PN 序列广泛应用于 CDMA 系统中，用于对传输数据进行扰码和解扰操作，以及用于直接序列扩频。PN 序列生成器的模块框图及其参数设置如图 4-25 所示。

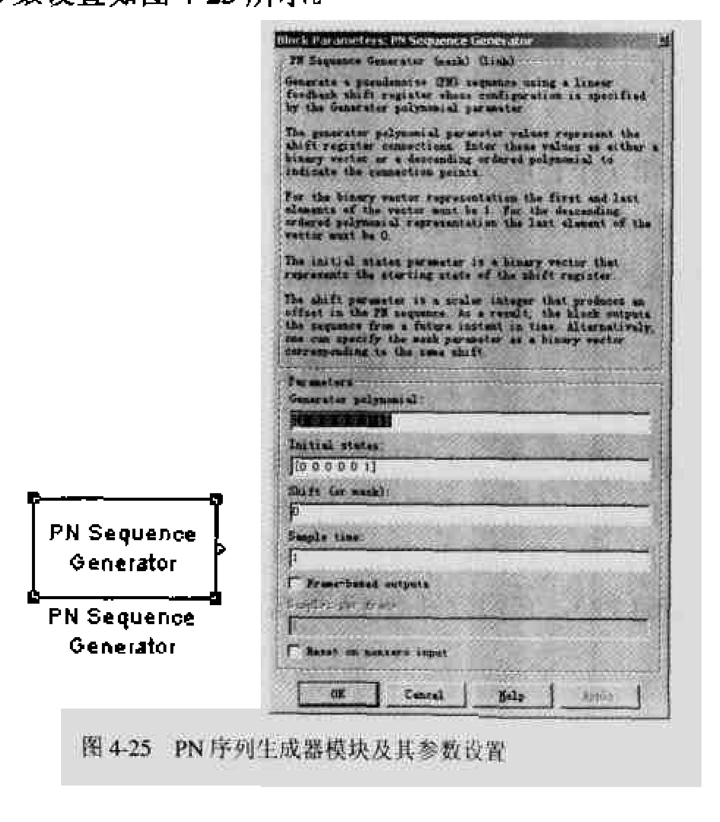



图 4-25　PN 序列生成器模块及其参数设置

PN 序列生成器采用移位寄存器来产生 PN 码，它的原理如图 4-28 所示。从图中可以看到，PN 序列生成器中共有 $r$ 个寄存器，每个寄存器都以相同的抽样频率更新寄存器的状态，即在第 $k$ 个寄存器在时刻 $t$+1 的状态 $m_k^{t+1}$ 等于第 $k$+1 个寄存器在时刻 $t$ 的状态 $m_{k+1}^t$ 。

PN 序列生成器可以表示为一个生成多项式

$$g_r z^r + g_{r-1} z^{r-1} + g_{r-2} z^{r-2} + \cdots + g_1 z + g_0 \tag{4.18}$$

其中 $g_i$ 对应于图 4-28 中的开关的状态：当 $g_i$ 等于 1 时表示开关 $g_i$ 闭合；当 $g_i$ 等于 0 时表示开关 $g_i$ 打开。PN 序列生成器的生成多项式有两种表示方式，一种方式是使用二进制向量 $\left[g_r, g_{r-1}, \cdots, g_0\right]$，另外一种方式是把生成多项式中 $g_i$ 不等于零的项的下标 $i$ 组成一个向量。例如，生成多项式 $z^8 + z^6 + z^2 + z + 1$ 可以表示成[1 0 1 0 0 0 1 1 1]，也可以表示成[8 6 2 1 0]。需要注意的时，在公式 4.18 中，系数 $g_r$ 和 $g_0$ 必须等于 1。

在产生输出序列之前，PN 序列生成器还允许对由生成多项式产生的序列实施移位或屏蔽操作，这种操作也可以用一个多项式表示，

$$\mathrm{m}_{r-1} z^{r-1} + \mathrm{m}_{r-2} z^{r-2} + \cdots + \mathrm{m}_1 z + \mathrm{m}_0 \tag{4.19}$$

其中的多项式系数 $\mathrm{m}_i$ 等于 0 或 1，分别对应于图 4-28 中开关 $\mathrm{m}_i$ 的打开或关闭状态。缺省状态下只有开关 $\mathrm{m}_0$ 是关闭的，这时候输出的是对应于开关 $\mathrm{m}_0$ 的寄存器的状态。由于多项式 $\mathrm{m}_k z^k$ 表示输出对应于开关 $\mathrm{m}_k$ 的寄存器的状态，它相对于寄存器 $\mathrm{m}_0$ 的时延等于 $k$，因此，在对输出序列实施移位操作时，只需把与公式 4.18 对应的参数设置为一个整数。如果需要对输出序列实施屏蔽操作，则应该按照公式 4.18 构造多项式。与生成多项式（公式 4.18）类似地，公式 4.19 也有两种表示方式。

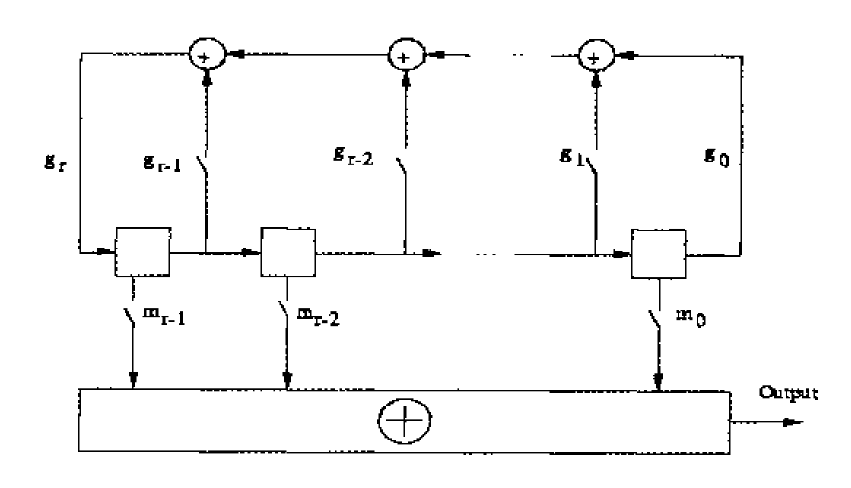



图 4-26　PN 序列的原理图

PN 序列生成器主要有以下几个参数。

■　Generator polynomial（生成多项式）

对应于公式 4.18 的生成多项式，既可以采用二进制向量表示，也可以采用由多项式下标构成的整数向量表示。

■ Initial states（初始状态）

PN 序列生成器中各个寄存器的初始状态，表示成一个二进制向量。

■ Shift (or mask)（时延或屏蔽操作）

对应于公式 4.19 的生成多项式。当它是一个标量时标是时延大小；当设置为向量时表示屏蔽操作的方式，这时候既可以采用二进制向量的方式，也可以使用由多项式下标构成的整数向量。

■ Sample time（抽样时间）

输出序列中每个元素的持续时间。

■ Frame-based outputs（帧格式输出）

指定 PN 序列生成器以帧格式产生输出序列。

■ Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■ Reset on nonzero input（复位方式）

当选择了 Reset on nonzero input 选项之后，PN 序列生成器提供了一个输入端口，用于输入复位信号。当复位信号输入端口不等于零时，PN 序列生成器将各个寄存器的状态恢复为初始状态（Initial states）。

在 PN 序列生成器中，只有当生成多项式是一个本原多项式（即生成多项式不可约）时才能够产生一个长度最大的 PN 序列。对于有 $n$ 个寄存器的 PN 序列生成器，它的最大长度等于 $2^n$-1。表 4-1 列出了寄存器长度不大于 53 时能够产生最长 PN 序列的一些生成多项式，以供参考。

**表 4-1　　能够产生最长 PN 序列的生成多项式**

| r | 生成多项式 | r | 生成多项式 | r | 生成多项式 | r | 生成多项式 |
|---|---|---|---|---|---|---|---|
| 2 | [2 1 0] | 15 | [15 14 0] | 28 | [28 25 0] | 41 | [41 3 0] |
| 3 | [3 2 0] | 16 | [16 15 13 4 0] | 29 | [29 27 0] | 42 | [42 23 22 1 0] |
| 4 | [4 3 0] | 17 | [17 14 0] | 30 | [30 29 28 7 0] | 43 | [43 6 4 3 0] |
| 5 | [5 3 0] | 18 | [18 11 0] | 31 | [31 28 0] | 44 | [44 6 5 2 0] |
| 6 | [6 5 0] | 19 | [19 18 17 14 0] | 32 | [32 31 30 10 0] | 45 | [45 4 3 1 0] |
| 7 | [7 6 0] | 20 | [20 17 0] | 33 | [33 20 0] | 46 | [46 21 10 1 0] |
| 8 | [8 6 5 4 0] | 21 | [21 19 0] | 34 | [34 15 14 1 0] | 47 | [47 14 0] |
| 9 | [9 5 0] | 22 | [22 21 0] | 35 | [35 2 0] | 48 | [48 28 27 1 0] |
| 10 | [10 7 0] | 23 | [23 18 0] | 36 | [36 11 0] | 49 | [49 9 0] |
| 11 | [11 9 0] | 24 | [24 23 22 17 0] | 37 | [37 12 10 2 0] | 50 | [50 4 3 2 0] |
| 12 | [12 11 8 6 0] | 25 | [25 22 0] | 38 | [38 6 5 1 0] | 51 | [51 6 3 1 0] |
| 13 | [13 12 10 9 0] | 26 | [26 25 24 20 0] | 39 | [39 8 0] | 52 | [52 3 0] |
| 14 | [14 13 8 4 0] | 27 | [27 26 25 22 0] | 40 | [40 5 4 3 0] | 53 | [53 6 2 1 0] |

### 4. Barker 码生成器

Barker 码是 PN 码的一个子集，具有很好的自相关性。MATLAB 中的 Barker 码生成器

（Barker Code Generator）能够产生长度不大于 13 的 Barker 码。表 4-2 是 Barker 码的序列。

表 4-2　Barker 码序列

| 码长 | 码序列（+1 、-1 序列） | 码序列（0、1 序列） |
|---|---|---|
| 1 | [-1] | 1 |
| 2 | [-1 +1] | 10 |
| 3 | [-1 -1 +1] | 110 |
| 4 | [-1 -1 +1 -1] | 1101 |
| 5 | [-1 -1 -1 +1 -1] | 11101 |
| 7 | [-1 -1 -1 +1 +1 -1 +1] | 1110010 |
| 11 | [-1 -1 -1 +1 +1 +1 -1 +1 +1 -1 +1] | 11100010010 |
| 13 | [-1 -1 -1 -1 -1 +1 +1 -1 -1 +1 -1 +1 -1] | 1111100110101 |

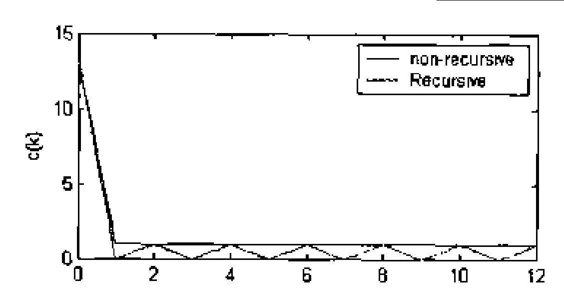



图 4-27　长度为 13 的 Barker 码的自相关特性

Barker 码具有很好的自相关特性，图 4-27 所示是长度为 13 的 Barker 码的自相关函数曲线。从图中可以看到，不管是采用循环方式还是非循环方式计算自相关函数，Barker 码只有在位移等于 0 时具有很强的相关性；当位移长度不等于 0 时，Barker 码的相关函数值都不超过 1，其中循环方式的自相关特性更好。Barker 码生成器及其参数设置对话框如图 4-28 所示。

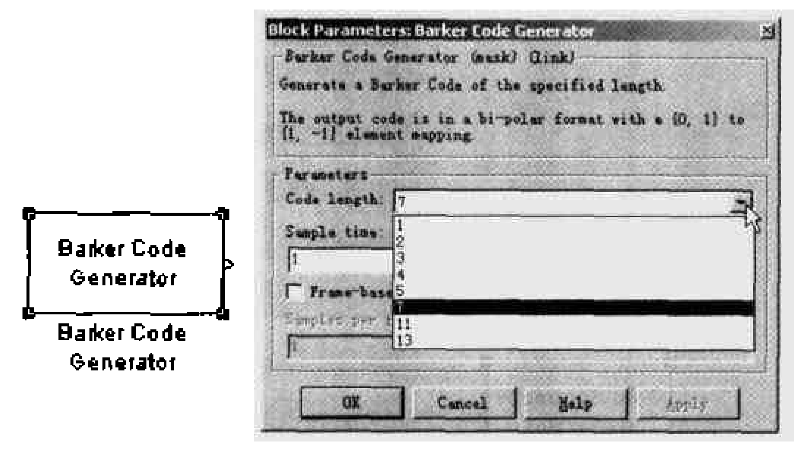



图 4-28　Barker 码生成器模块及其参数设置

Barker 码生成器主要有以下几个参数。

■　Code length（码长）

Barker 码的长度。可选的长度是 1、2、3、4、5、7、11 和 13。

■　Sample time（抽样时间）

输出序列中每个元素的持续时间。

■ Frame-based outputs（帧格式输出）

指定 Barker 码生成器以帧格式产生输出序列。

■ Samples per frame（每帧的抽样数）

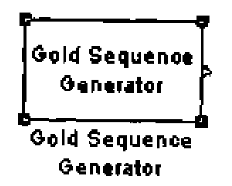
当选择了"Frame-based outputs"参数之后，本参数用来确定每帧的抽样点的数目。

### 5. Gold 序列生成器

Gold 序列生成器（Gold Sequence Generator）产生一个 Gold 序列。Gold 序列的一个重要特性是它具有很好的互相关特性，它根据两个长度 $N=2^n-1$ 的序列 $u$ 和 $v$ 产生一个 Gold 序列 $G(u,v)$，这两个序列称为一个"优选对"（Preferred Pair）。Gold 序列 $G(u,v)$可以用公式 4.20 表示：

$$G(u,v)=\{u,v,u\oplus v,u\oplus Tv,u\oplus T^2v,\cdots,u\oplus T^{N-1}v\} \tag{4.20}$$

其中，$T^n x$ 表示将序列 $x$ 以循环移位的方式向左移 $n$ 位，$\oplus$ 表示模二加法。从公式 4.20 可以看出，由长度为 $N$ 的两个序列 $u$ 和 $v$ 产生的 Gold 序列 $G(u,v)$中包含了 $N+2$ 个长度为 $N$ 的序列，Gold 序列生成器可以根据设定的参数输出其中的某个序列。Gold 序列生成器如图 4-29 所示。



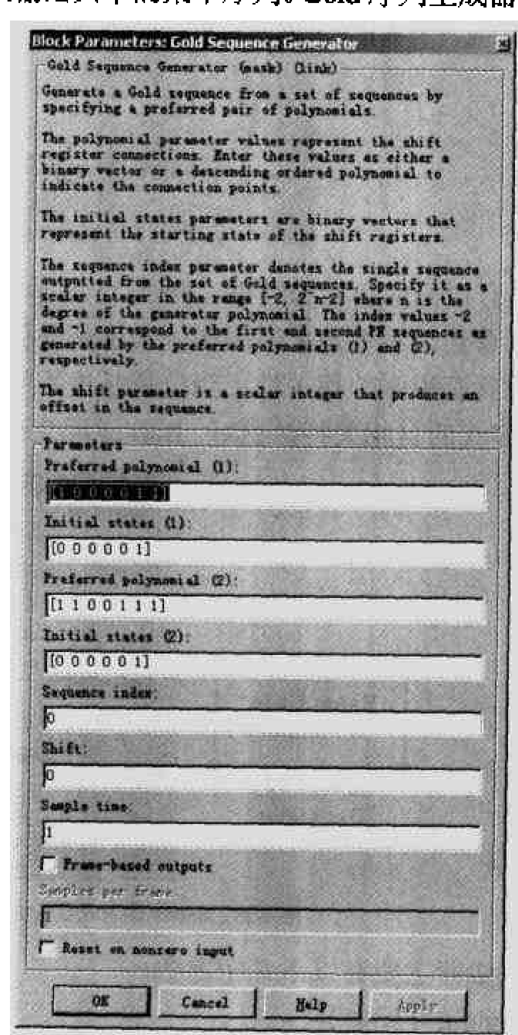




图 4-29 Gold 序列生成器模块及其参数设置

Gold 码实际上是把两个相同长度的 PN 序列产生器产生的 PN 序列进行异或运算后得到的序列，如图 4-30 所示。

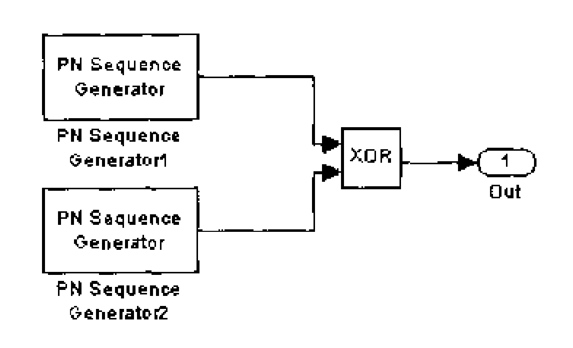



图 4-30　Gold 码生成器的结构

在图 4-30 所示中，两个 PN 序列产生器产生的 PN 序列必须具有某种特性（即符合“优选对”的要求）才能够产生 Gold 码。如果两个序列 $X$ 和 $Y$ 是长度为 $N = 2^n - 1$ 优选对，则 $n$ 不能被 4 整除，同时 $Y = X[q]$，即序列 $Y$ 是通过对序列 $X$ 每隔 $q$ 个元素进行一次抽样得到的序列，其中 $q$ 是一个奇数，$q = 2^k + 1$ 或 $q = 2^{2k} - 2^k + 1$，并且 $n$ 和 $k$ 的最大公约数下面的满足条件：

$$\gcd(n,k) = \begin{cases} 1, n \equiv 1 \bmod 2 \\ 2, n \equiv 2 \bmod 4 \end{cases}$$

表 4-3 列出了一些常用优选对的生成多项式。

**表 4-3　一些常用优选对的生成多项式**

| n | N | 优选对序列 1 | 优选对序列 2 |
|---|---|---|---|
| 5 | 31 | [5 2 0] | [5 4 3 2 0] |
| 6 | 63 | [6 1 0] | [6 5 2 1 0] |
| 7 | 127 | [7 3 0] | [7 3 2 1 0] |
| 9 | 511 | [9 4 0] | [9 6 4 3 0] |
| 10 | 1023 | [10 3 0] | [10 8 3 2 0] |
| 11 | 2047 | [11 2 0] | [11 8 5 2 0] |

设 $X$ 和 $Y$ 两个 Gold 码，它们属于同一个集合 $G(u, v)$，且长度等于 $2^n - 1$，则这两个序列的互相关函数 $R_k(X,Y)$只可能取 3 种值：$-t(n)$，$-1$ 以及 $t(n)-2$，其中：

$$t(n) = \begin{cases} 1 + 2^{\frac{n+1}{2}}, n\text{是奇数} \\ 1 + 2^{\frac{n+2}{2}}, n\text{是偶数} \end{cases}$$

图 4-33 所示是一个 Gold 码互相关函数曲线图。这里我们使用了表 4-5 中 $n$ 等于 5 时的优选对，并且两个优选对序列的初始状态都等于[0 0 0 0 1]，并且以序列号为 0 和 17 的两个序列为基础计算得到的。根据 $n$ 等于 5 可以算出 $t(n) = 9$，因此互相关函数 $R_k(X,Y)$只可能取 3 种值：-9、-1 和 7。图 4-31 所示的结果也印证了这一点。

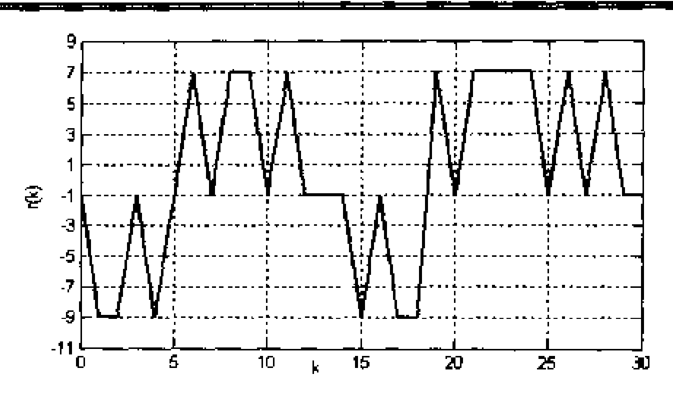



图 4-31 Gold 码的一个互相关函数

Gold 码生成器主要有以下几个参数。

■ Preferred polynomial（1）(优选对序列 1 的生成多项式)

优选对序列 1 的生成多项式，可以是二进制向量的形式，也可以是由多项式下标构成的整数向量。

■ Initial states（1）(优选对序列 1 的初始状态)

优选对序列 1 的初始状态。它是一个二进制向量，用于表明与优选对序列 1 对应的 PN 序列生成器中每个寄存器的初始状态。

■ Preferred polynomial（2）(优选对序列 2 的生成多项式)

优选对序列 2 的生成多项式，可以是二进制向量的形式，也可以是由多项式下标构成的整数向量。

■ Initial states（2）(优选对序列 2 的初始状态)

优选对序列 2 的初始状态。它是一个二进制向量，用于表明与优选对序列 2 对应的 PN 序列生成器中每个寄存器的初始状态。

■ Sequence index（序列号）

由两个优选对序列 $u$ 和 $v$ 生成的 $G(u, v)$是一个 Gold 码集合，参数 Sequence index 用于指定其中的一个序列作为输出序列。表 4-4 是 Sequence index 与 $G(u, v)$中元素的对应关系。

表 4-4 参数 Sequence index 与 $G(u, v)$中元素的对应关系

| Sequence index | 输出序列 |
|---|---|
| -2 | $u$ |
| -1 | $v$ |
| 0 | $u \oplus v$ |
| 1 | $u \oplus Tv$ |
| 2 | $u \oplus T^2 v$ |
| … | … |
| $2^n$-2 | $u \oplus T^{2^n-2} v$ |

■ Shift（时延）

指定 Gold 码生成器的输出序列的时延。该参数是一个整数，表示序列延迟 Shift 个抽样周期后输出。

■　Sample time（抽样时间）

输出序列中每个元素的持续时间。

■　Frame-based outputs（帧格式输出）

指定 Gold 码生成器以帧格式产生输出序列。

■　Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■　Reset on nonzero input（复位方式）

当选择了 Reset on nonzero input 选项之后，Gold 码生成器提供了一个输入端口，用于输入复位信号。当复位信号输入端口不等于零时，Gold 码生成器将各个寄存器的状态恢复为初始状态（Initial states[1]和 Initial states[2]）。

### 6. OVSF 码生成器

OVSF 码是在第三代移动通信系统中引入的一种码，用于保持不同信道之间的正交性。OVSF 的全称是 Orthogonal Variable Spreading Factor，可直译为“正交可变扩频因子”。OVSF 码与相同长度的 Hadamard 矩阵的行向量是相同的，但是它们的下标并不一致。OVSF 码可以看作是对码序列正交性的另外一种阐述方式。在 MATLAB 中，OVSF 码生成器（OVSF Code Generator）用于产生 OVSF 码，其模块框图和参数设置如图 4-32 所示。

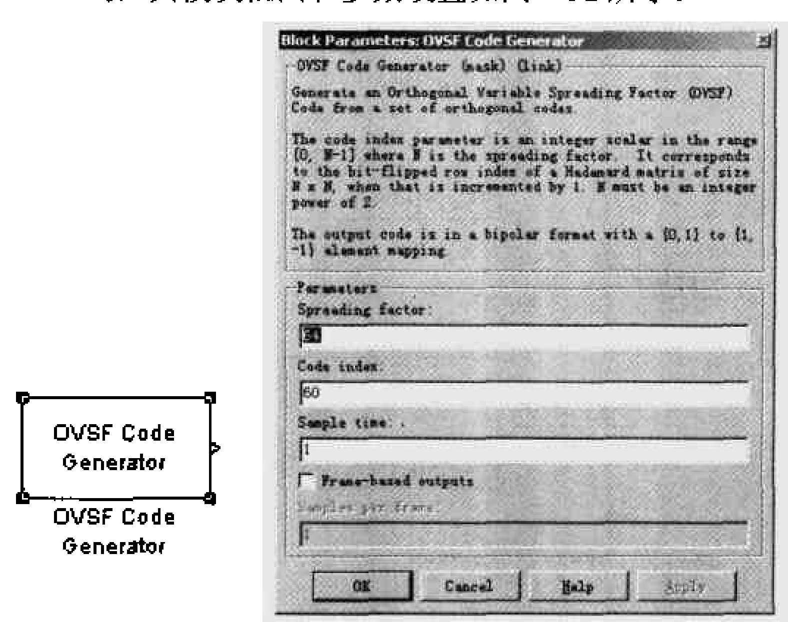



图 4-32　OVSF 码生成器模块及其参数设置

OVSF 码定义为一个 $N$ 行 $N$ 列矩阵 $C_N$ 的行向量，其中 $N=2^n$。矩阵 $C_N$ 可以用递归的方式定义，如公式 4.21 所示。

$$\begin{cases} C_1=[1] \\ C_{2N}=\begin{bmatrix} C_N(0) & C_N(0) \\ C_N(0) & -C_N(0) \\ C_N(1) & C_N(1) \\ C_N(1) & -C_N(1) \\ \cdots & \cdots \\ C_N(N-1) & C_N(N-1) \\ C_N(N-1) & -C_N(N-1) \end{bmatrix} \end{cases} \tag{4.21}$$

其中 $C_N(k)$表示矩阵 $C_N$ 的第 $k$ 行。

OVSF 码也可以用如图 4-35 所示的二叉树来表示。在这个二叉树中，根结点是$C_1(0)$。根据公式 4.21，由$C_1(0)$产生的矩阵：

$$C_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

矩阵的两个行向量就是根结点$C_1(0)$的两个子节点。同理，$C_2$的两个行向量$C_2(0)$和$C_2(1)$分别产生两个子节点$C_4(0)$、$C_4(1)$以及$C_4(2)$、$C_4(3)$。以此类推，可以构造一棵完整的二叉树。如果假定根结点$C_1(0)$处于第 0 层，则处于第 $i$ 层的节点是一个长度为 $2^i$ 的序列。

关于 OVSF 码产生的二叉树的一个特点是，对于处于二叉树某一层的一个节点 $C$，从跟节点到节点 $C$ 路径上的所有节点以及节点 $C$ 的子节点（包括子节点的子节点）不能与 $C$ 保持正交，而除此之外的所有节点都和 $C$ 正交。例如，在通信系统中，如果某个信道选取了节点$C_4(1)$对应的序列作为扩频码，由于节点$C_1(0)$、$C_2(0)$、$C_8(2)$以及$C_8(3)$都不与$C_4(1)$正交，为了在通信系统中保持原有的正交性，这 4 个节点对应的序列都不应该再分配给其他的信道。图 4-33 所示为 OVSF 码的树状表示。

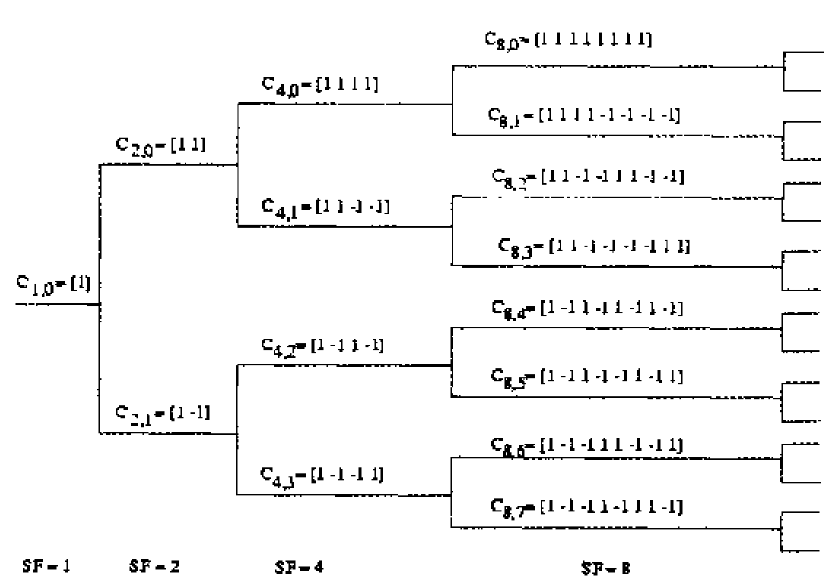



图 4-33　OVSF 码的树状表示

OVSF 码生成器主要有以下几个参数。

■ Spreading factor（扩频因子）

扩频因子用于确定 OVSF 码的长度，它等于 2 的 $n$ 次幂。

■ Code index（码序号）

当扩频因子等于 $N$ 时，码序号是处于 0 和 $N-1$ 之间的整数，并且选取矩阵 $C_N$ 的第 Code index 个行向量作为输出。

■ Sample time（抽样时间）

输出序列中每个元素的持续时间。

■ Frame-based outputs（帧格式输出）

指定 OVSF 码生成器以帧格式产生输出序列。

■ Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

### 7. Kasami 序列生成器

Kasami 序列是一组长度 $N=2^n-1$ 的具有很好的互相关特性的序列，其中 $n$ 是一个非负偶数。Kasami 序列分为两种类型：小集合（Small Set）和大集合（Large Set），小集合是大集合的子集，但是小集合的互相关函数具有更好的特性。

设 $u$ 是一个长度为 $N$ 的二进制序列，对序列 $u$ 每隔 $2^{n/2}+1$ 个元素进行一次抽样，得到序列 $w$，则公式 4.22 表示 Kasami 序列的小集合。

$$K_S(u,n,m)=\begin{cases}u, m=-1\\ u\oplus T^m w, m=0,\cdots,2^{n/2}-2\end{cases} \tag{4.22}$$

其中 $T^m w$ 表示把序列 $w$ 左移 $m$ 位，$\oplus$ 表示模二加法。从公式中可以看出，Kasami 序列的小集合共有 $2^{n/2}$ 个 Kasami 序列。

再次对序列 $u$ 每隔 $2^{n/2+1}+1$ 个元素进行一次抽样，由此得到序列 $v$。当 $n$ 满足条件 $n\equiv 2\bmod 4$ 时，Kasami 序列的大集合定义为公式 4.23 所示。

$$K_L(u,n,k,m)=\begin{cases}u, k=-2, m=-1\\ v, k=-1, m=-1\\ u\oplus T^k v, k=0,\cdots,2^n-2, m=-1\\ u\oplus T^m w, k=-2, m=0,\cdots,2^{n/2}-2\\ v\oplus T^m w, k=-1, m=0,\cdots,2^{n/2}-2\\ u\oplus T^k v\oplus T^m w, k=0,\cdots,2^n-2, m=0,\cdots,2^{n/2}-2\end{cases} \tag{4.23}$$

对比公式 4.23 和公式 4.20，可以看到，Kasami 序列大集合的前三行与 Gold 码的定义是一致的，Kasami 序列实际上是 Gold 码的一个超集。表 4-5 列出了一些能够产生 Kasami 序列的 $u$ 的生成多项式。

**表 4-5　一些能够产生 Kasami 序列的 $u$ 的生成多项式**

| $n$ | $N$ | 序列 $u$ 的生成多项式 | Kasami 序列的类型 |
|---|---|---|---|
| 4 | 15 | [4 1 0] | 小集合 |
| 6 | 63 | [6 1 0] | 大集合 |
| 8 | 255 | [8 4 3 2 0] | 小集合 |
| 10 | 1023 | [10 3 0] | 大集合 |
| 12 | 4095 | [12 6 4 1 0] | 小集合 |

Kasami 序列的互相关函数的取值范围是$\{-t(n),-s(n),-1, s(n)-2, t(n)-2\}$，其中 $t(n)=1+2^{(n+2)/2}$（$n$ 是偶数），$s(n)=(t(n)+1)/2$。图 4-34 所示是一个长度为 63 的 Kasami 序列的互相关函数，它采用 $n=6$ 的 Kasami 序列大集合，并且[$k$ $m$]分别等于[2 2]和[3 4]。根据前面的计算公式可以得到 $t(n)$ 等于 17，$s(n)$等于 9，因此长度等于 63 的 Kasami 序列的互相关函数的取值范围是$\{-17,-9,-1,7,15\}$，这与图 4-34 所示的仿真结果相吻合。

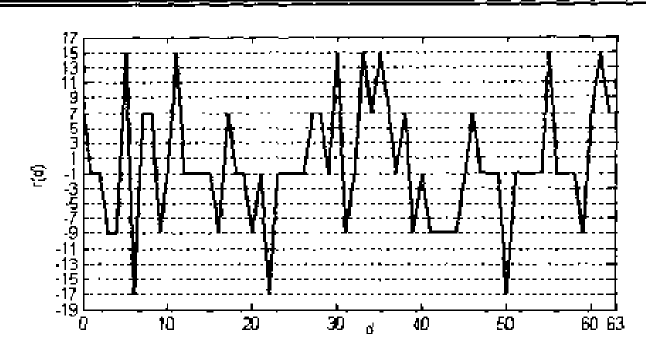


图 4-34 长度等于 63 的 Kasami 序列的互相关函数

图 4-35 所示是 Kasami 序列生成器模块及其参数设置对话框。

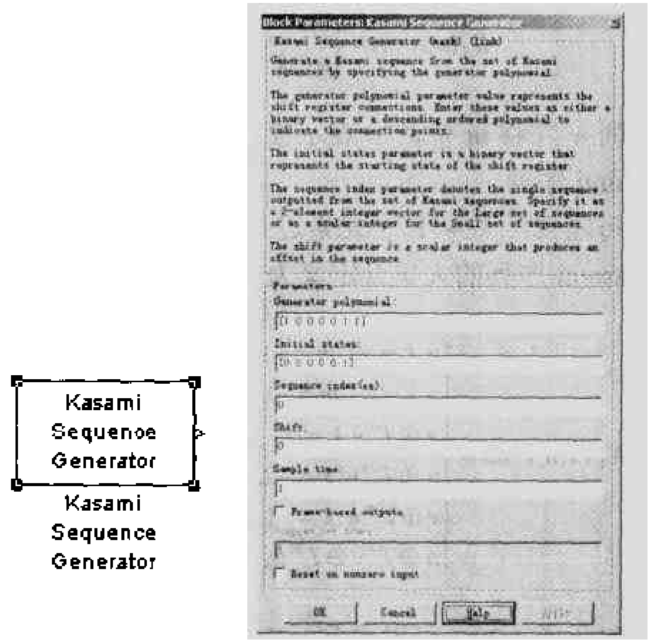



图 4-35 Kasami 序列生成器模块及其参数设置

Kasami 序列生成器主要有以下几个参数。

■ Generator polynomial（生成多项式）

序列 $u$ 的生成多项式，它可以用多项式的系数组成的向量来表示，也可以用多项式的指数组成的向量来表示。

■ Initial states（初始状态）

用于产生序列 $u$ 的寄存器的初始状态，它是一个向量，向量的长度等于数列 $u$ 的生成多项式的最大指数。

■ Sequence index（码序号）

用于指定 Kasami 序列的序号，Kasami 序列小集合和大集合中码序号的编排方式分别如表 4-6 和表 4-7 所示。

表 4-6 Kasami 序列小集合中码序号的编排方式

| Sequence index | 参数 m 的取值范围 | 输出序列 |
| --- | --- | --- |
| −1 | $m=-1$ | $u$ |
| $m$ | $m=0, 1, \ldots, 2^{n/2}-2$ | $u \oplus T^{m} w$ |

表 4-7　　Kasami 序列大集合中码序号的编排方式

| Sequence index[k m] | 参数 k 和 m 的取值范围 | 输出序列 |
|---|---|---|
| [−2 −1] | $k=-2, m=-1$ | $u$ |
| [−1 −1] | $k=-1, m=-1$ | $v$ |
| [k −1] | $k=0, 1, \ldots, 2^{n/2}-2$<br>m = −1 | $u \oplus T^{k}v$ |
| [−2 m] | $k=-2$<br>$m=0, 1, \ldots, 2^{n/2}-2$ | $u \oplus T^{m}w$ |
| [−1 m] | $k=-1$<br>$m=0, 1, \ldots, 2^{n/2}-2$ | $v \oplus T^{m}w$ |
| [k m] | $k=0, 1, \ldots, 2^{n/2}-2$<br>$m=0, 1, \ldots, 2^{n/2}-2$ | $u \oplus T^{k}v \oplus T^{m}w$ |

■　Shift（时延）

指定 Kasami 序列生成器的输出序列的时延。该参数是一个整数，表示序列延迟 Shift 个抽样周期后输出。

■　Sample time（抽样时间）

输出序列中每个元素的持续时间。

■　Frame-based outputs（帧格式输出）

指定 Kasami 序列生成器以帧格式产生输出序列。

■　Samples per frame（每帧的抽样数）

当选择了 Frame-based outputs 参数之后，本参数用来确定每帧的抽样点的数目。

■　Reset on nonzero input（复位方式）

当选择了 Reset on nonzero input 选项之后，Kasami 序列生成器提供了一个输入端口，用于输入复位信号。当复位信号输入端口不等于零时，Kasami 序列生成器将各个寄存器的状态恢复为初始状态。

## 4.1.6　实例 4.1——通过压控振荡器实现 BFSK 调制

在移动通信系统中，二进制频移键控 BFSK（Binary Frequency Shift Keying）是最早使用的一种调制方式。BFSK 的调制和解调过程比较简单，第一代移动通信网络，如美国的 AMPS，采用的就是 BFSK 调制方式。由于压控振荡器能够根据输入信号的大小产生不同的输出频率，因此本实例采用压控振荡器来实现 BFSK 调制，其结构框图如图 4-36 所示。

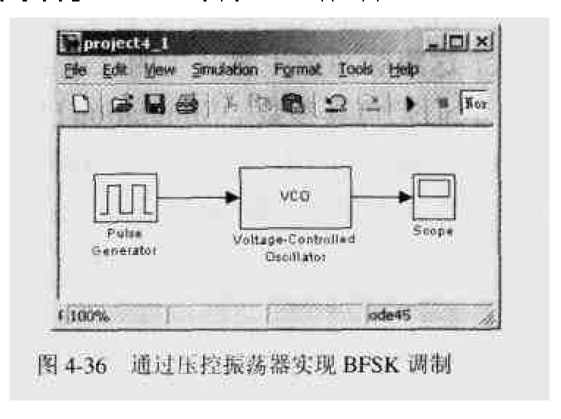



图 4-36　通过压控振荡器实现 BFSK 调制

本实例采用 Pulse Generator（脉冲产生器）产生一个周期为 2 秒、占空比为 50%的脉冲信号，用这个脉冲信号驱动压控振荡器，使输出信号的频率随着脉冲产生器的变化而变化，从而实现 BFSK 调制。Scope（示波器）用来显示 BFSK 信号的波形。

表 4-8 所示是 Pulse Generator（脉冲产生器）的参数设置，其中输出信号的幅度为 1，振荡频率为 10（单位：Hz），相位延迟为 0。由于脉冲产生器产生的脉冲的占空比是 50%，因此脉冲产生器的输出就等同于一个 0 和 1 交替的二进制序列 101010…，序列中每个符号的周期等于 1 秒。

表 4-9 所示是 Voltage-Controlled Oscillator（压控振荡器）的参数设置。在本实例中，我们采用的是连续时间的压控振荡器，输出信号的幅度为 1，振荡频率为 10 Hz，初始相位为 0。另外，我们设置输入信号灵敏度为 10，使得输入信号的幅度变化 1 V 时，输出信号的频率变化是 10 Hz。这样，当输入信号为 0 时，输出信号的频率是 10 Hz；当输入信号为 1 时，输出信号的频率是 20 Hz。这种输入信号的频率变化就等同于 BFSK 调制。

**表 4-8　Pulse Generator（脉冲产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Pulse Generator |
| Amplitude | 1 |
| Period (secs) | 2 |
| Pulse Width (% of period) | 50 |
| Phase Delay (secs) | 0 |
| Interpret vector parameters as 1-D | Checked |

**表 4-9　Voltage-Controlled Oscillator（压控振荡器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Voltage-Controlled Oscillator |
| Output amplitude | 1 |
| Oscillation frequency (Hz) | 10 |
| Input sensitivity | 10 |
| Initial phase (rad) | 0 |

图 4-37 所示是本实例产生的 BFSK 信号的波形。从图 4-37 所示中可以看到，输出信号的频率每隔 1 秒变化一次，这跟脉冲产生器产生的二进制交替信号的周期是一致的。值得注意的是，在本实例中，我们把仿真参数“Simulation | Simulation Parameters… | Solver | Output Options | Refine factor”设置为 10，以此提高 MATLAB 对仿真结果的抽样频率。否则，如果使用缺省值 1，示波器中显示的将是一条水平直线，观察不到输出信号频率的变化。



图 4-37 实例 4.1 产生的 BFSK 波形

# 4.2 信宿

信宿是通信系统的终点，它要对接收到的信号进行处理，通过解调和解码等过程去除其中的干扰，最后得到发送端的原始数据。在通信系统的仿真过程中，用户希望能够把接收到的数据通过某种方式保存或显示出来，以直观的形态对仿真的结果进行评估。MATLAB 提供了若干个模块用于实现这种功能，在本节里我们将依次介绍这几种模块。

## 4.2.1 示波器

示波器（Scope）是最常用的一种输出工具，它把信号按照时间顺序在二维坐标轴上显示出来。示波器模块及其显示界面如图 4-38 所示。

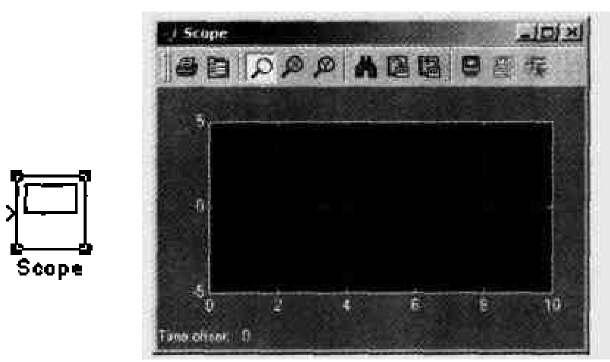


图 4-38 示波器模块及其显示界面

从图 4-38 所示中可以看到，示波器的横坐标表示时间轴，纵坐标表示信号的强度。在缺省情况下，横坐标的坐标范围是[0, 10]，纵坐标的范围是[−5, 5]。仿真过程中我们可以按照需要改变坐标轴的设置。

在示波器的显示界面上单击鼠标右键，从弹出式菜单中选择“Axes properties…”选项，出现如图 4-39 所示的对话框，其中可以设置纵坐标的最小值 Y-min 和最大值 Y-max，以及纵坐标的标题 Title。

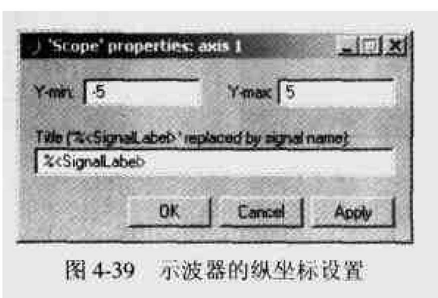


图 4-39 示波器的纵坐标设置

在示波器显示界面的工具栏中单击第二个图标后，出现一个参数设置窗口，这个窗口由两个参数设置页，如图 4-40 所示。



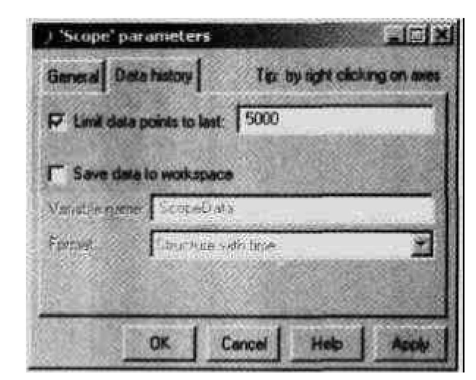




图 4-40　示波器的两个参数设置页

图 4-40 所示的各个参数分别说明如下。

■　Number of axes（纵坐标的个数）

缺省情况下，纵坐标的个数是 1，这时候只有一个坐标图；当纵坐标的个数大于 1 时，示波器划分成多个坐标，并且示波器有多个输入端口，这些坐标有不同的纵坐标，但是它们的横坐标是相同的。

■　Time range（时间轴的显示范围）

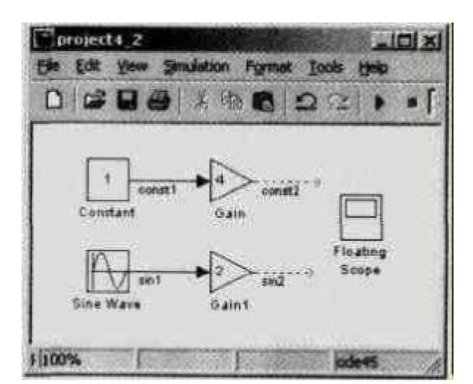
设置示波器的时间轴的最大显示范围，单位是秒。当设置为 auto 时，时间轴的显示范围就等于整个仿真时间段。

■　Tick labels（坐标轴的显示标签）

确定示波器坐标轴上标签的显示方式：当选择“all”时，显示所有纵坐标和所有横坐标的标签；当选择“none”时，不显示任何坐标轴的标签；当选择“bottom axis only”时，只显示各个纵坐标以及最下面的横坐标的标签。

■　Floating scope（浮动示波器）

当选择该选项时，示波器将变成一个浮动示波器。浮动示波器是一个可以方便的改变输入信号的示波器，它没有输入端口，通过动态的选择输入信号，可以方便地在多个输入信号之间切换。图 4-41 所示是关于浮动示波器的一个实例程序的模块框图及其运行结果。



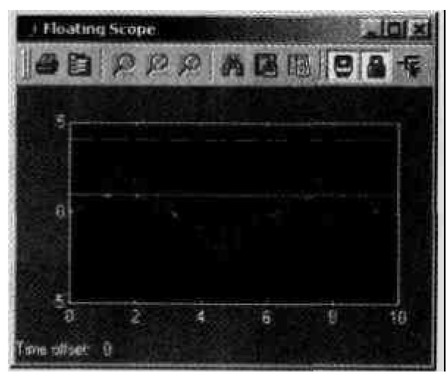




图 4-41　一个浮动示波器的示例

在这个例子中有两个信号产生器 Constant 和 Sine Wave，分别产生恒定信号 const1 和正弦波信号 sin1。这两个信号通过增益放大器 Gain 和 Gain1 后得到信号 const2 和 sin2。浮动示

波器虽然没有与这 4 个信号之间相连，但是能够使示波器在这几个信号之间进行切换。浮动示波器的切换是通过如图 4-42 所示的信号选择对话框实现的。图 4-41 所示的示波器输出选择了 4 个信号。

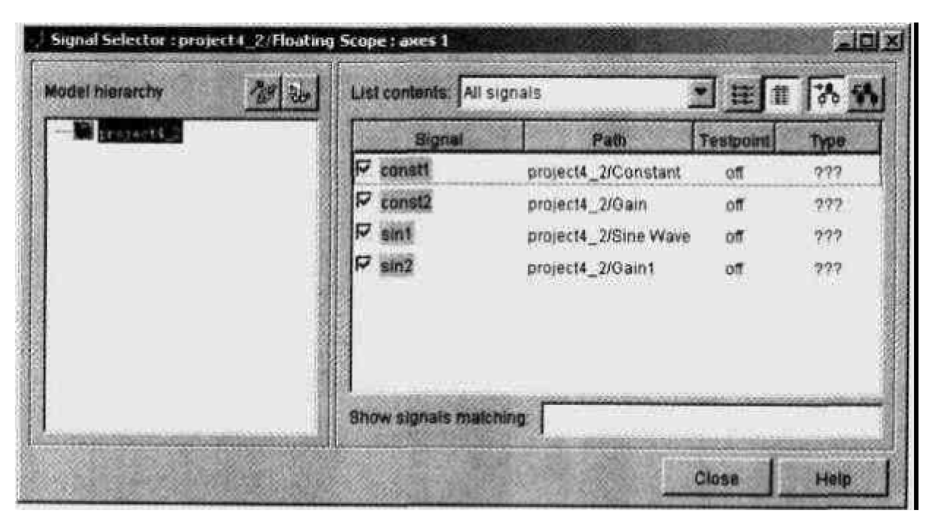



图 4-42　浮动示波器的信号选择

需要注意的是，在这个实例程序中，还需要修改以下运行参数，将"Simulation | Simulation parameters... | Advanced | Optimizations | Signal storage reuse" 设置为 off。

■　Sampling（显示方式）

本参数用于确定示波器的显示方式。当选择抽取方式（Decimation）时，示波器将每隔若干个输入信号显示一个输出；当选择采样方式（Sampling）时，示波器每隔一个采样时间产生一个输出。

■　Limit data points to last（数据储存个数）

指定示波器能够储存的数据的最大数目 $N$。当输入数据的数目大于这个最大值 $N$ 时，示波器只能显示最后 $N$ 个数据。如果要显示所有的数据，取消这个复选框。

■　Save data to workspace（保存数据到工作区）

当选择这个复选框之后，示波器除了显示输入的数据之外，还将这些数据保存到指定的工作区中。

■　Variable name（工作区变量名）

当选择了 Save data to workspace 之后，本参数用于指定工作区变量的名称。

■　Format（数据保存的格式）

当选择了"Save data to workspace"之后，本参数用于指定数据在工作区中的保存格式。数据保存格式有 3 种：数组格式（Array）、结构方式（Structure）以及带时间的结构方式（Structure with time）。数组格式适用于只有一个输入变量的示波器；后两种格式适用于有多个输入信号的示波器，两者的差别在于，前者保存的数据中不包含时间信息，而后者还包含了与每个数据对应的时间。

## 4.2.2　错误率统计

错误率统计模块（Error Rate Calculation）分别从发送端和接收端得到输入数据，对这两

个输入数据进行比较，并且根据比较的结果计算错误率。如果输入信号是二进制数据，则统计的结果是误比特率，否则，统计得到的结果是错误符号率。值得注意的是，错误率统计模块只比较两个输入信号的正负关系，而不具体地比较它们的大小。错误率统计模块及其参数设置如图 4-43 所示。

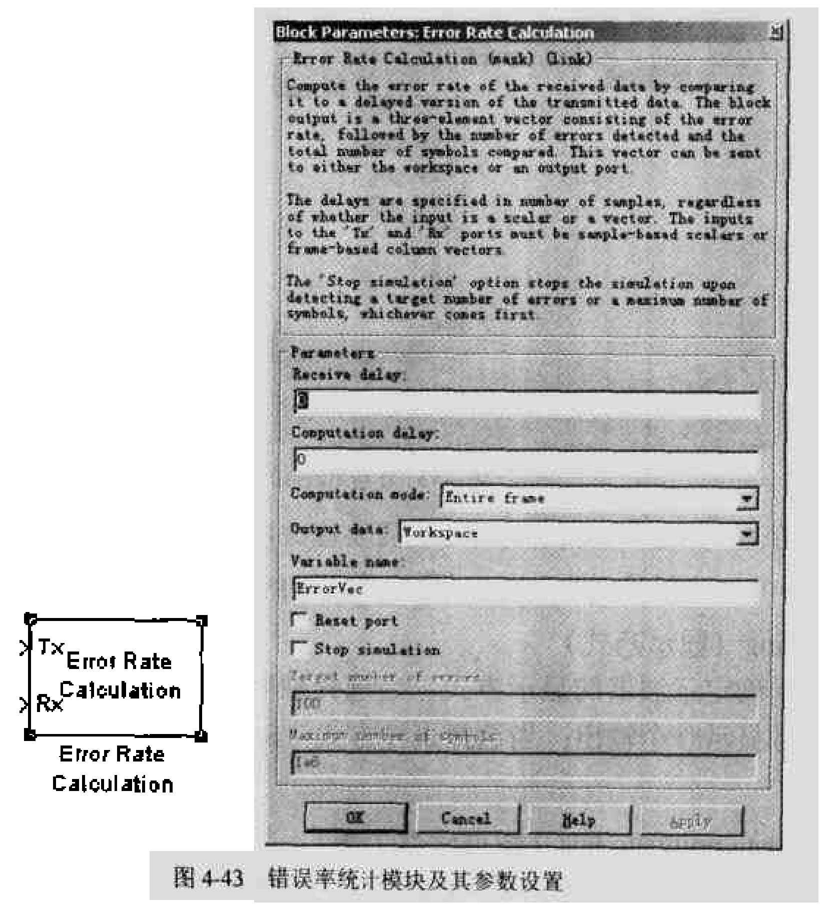



图 4-43 错误率统计模块及其参数设置

错误率统计模块主要有以下几个参数。

■ Receive delay（接收端时延）

在通信系统中，接收端需要对接收到的信号进行解调、解码或解交织，这些过程可能会带来一定的时延，使得到达错误率统计模块接收端口的信号滞后于发送端口的信号。为了弥补这种时延，错误率统计模块需要把发送端的输入数据延迟若干个输入数据，参数 Receive delay 表示接收端输入的数据滞后发送端数据若干个输入数据。

■ Computation delay（计算时延）

在通信系统的仿真过程中，有时候需要忽略最初的若干个输入数据，这是通过计算时延这个参数实现的。计算时延指示错误率统计模块忽略最初的若干个输入数据。

■ Computation mode（计算模式）

错误率统计模块有 3 种计算模式：帧计算模式（Entire frame），错误率统计模块对发送端和接受端的所有输入数据进行统计；掩码模式（Select samples from mask），错误率统计模块根据掩码指定对特定的输入数据进行统计，掩码的内容由参数 Selected samples from frame 确定；端口模式（Select samples from port），这时候错误率统计模块增加一个输入端口 Sel，只有这个端口的输入信号有效时才统计错误率。

■ Selected samples from frame（掩码）

当计算模式（Computation mode）设置为 Select samples from mask.时，本参数用于确定哪些输入数据需要实施统计。例如，当设置为[1 5]时，错误率统计模块只对每帧的第一个和第五个输入数据进行统计，同时忽略其他的数据。

■ Output data（输出数据）

指定输出数据的方式：Workspace 表示将统计数据输出到工作区，Port 则把统计数据从端口中输出。

■ Variable name（变量名）

当输出数据（Output data）设置为 Workspace 时，本参数指定了用于保存统计数据的工作区变量的名称。

■ Reset port 复位端口

当选择了复位端口（Reset port）时，错误率统计模块增加一个输入端口 Rst，当这个信号有效时，错误率统计模块被复位，统计值重新设置为 0。

■ Stop simulation（停止仿真条件）

当选择了本参数之后，如果错误率统计模块检测到指定数目的错误，或者数据的比较次数达到某个门限，则停止仿真过程。

■ Target number of errors（错误门限）

当选择了 Stop simulation 参数后，本参数用于指定在仿真停止之前允许出现的错误的最大个数。

■ Maximum number of symbols（比较门限）

当选择了 Stop simulation 参数后，本参数用于指定在仿真停止之前允许比较的输入数据的最大个数。

### 4.2.3 将结果输出到文件

MATLAB 提供了两种仿真模块，用于把仿真结果保存到文件中，这两个模块分别是文件写入（To File）模块和触发式文件写入（Triggered Write To File）模块。这两个模块之间的差别在于，后者是“触发式”的，它只在输入信号的上升沿有效的情况下才把仿真结果写入到文件中。

#### 1. 文件写入模块

文件写入模块（To File）把仿真数据写入指定路径的指定文件中，如图 4-44 所示。文件读取模块的文件路径和名字由参数 File name 确定，并且这个文件组织成一个（$m, n$）矩阵的形式。这个矩阵的结构与 4.1.2 节中介绍的文件读取模块的矩阵结构是相同的。

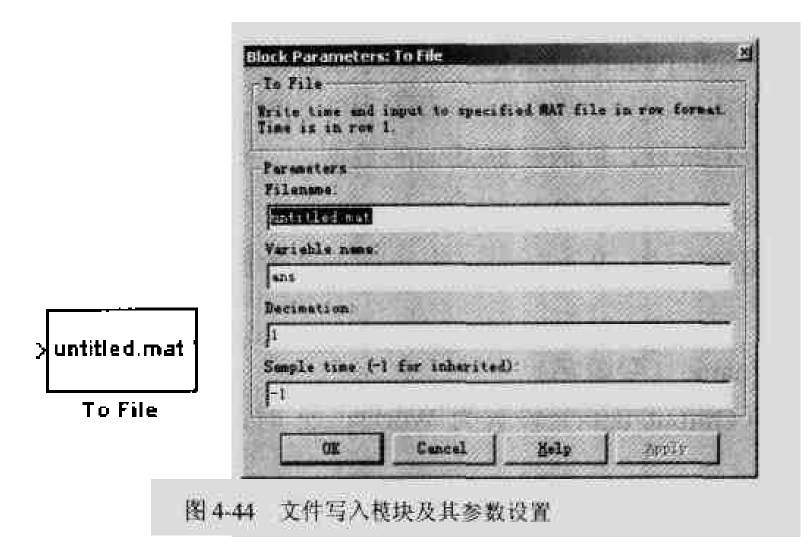



图 4-44　文件写入模块及其参数设置

文件写入模块主要有以下几个参数。

■　File name（文件名称）

文件的完整名称，包括文件的扩展名及其所在的路径。

■　Variable name（变量名称）

保存在文件中的矩阵的变量名。

■　Decimation（抽取因子）

文件写入模块可以以一定的抽样间隔保存仿真结果。抽取因子的缺省值是 1，表示把每一个仿真结果都保存到文件中。当抽取因子等于 $n$ 时，文件写入模块每隔 $n$ 个输入数据把最后一个数据保存到文件中。

■　Sample time（抽样时间）

文件写入模块的抽样间隔。

### 2. 触发式文件写入模块

触发式文件写入模块（Triggered Write To File）在触发信号上升沿到来时把输入的数据保存到指定的文件中。触发式文件写入模块及其参数设置如图 4-45 所示。

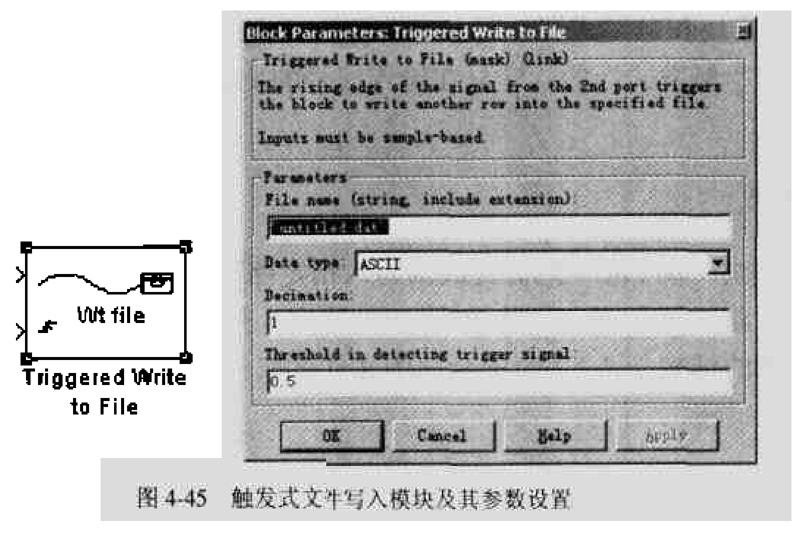



图 4-45　触发式文件写入模块及其参数设置

触发式文件写入模块的参数主要有以下几个。

■　File name（文件名称）

文件的完整名称，包括文件的扩展名及其所在的路径。

■　Data type（数据类型）

触发式文件读取模块可以读取 4 种类型的数据，即字符型（ASCII）、二进制型（Binary）、浮点型（Float）以及整型（Integer）。触发式文件写入模块把输入数据转化成相应的类型写入到指定的文件中。

■　Decimation（抽取率）

抽取率等于 1 时，触发式文件写入模块在每个输入信号的上升沿都执行一次写操作；当抽取率等于 $n$ 时，触发式文件读取模块每隔 $n$ 个上升沿执行一次文件写入操作。不管抽取率是否大于 1，触发式文件读取模块在输入信号的第一个上升沿都会写一次文件。

■　Threshold in detecting trigger signal（触发门限）

触发信号的门限。当输入信号的强度由低至高达到触发门限时，MATLAB 认为出现了一个上升沿。

### 3. 将结果保存到工作区

工作区写入模块（To Workspace）把数据保存到工作区中。工作区写入模块及其参数设置如图 4-46 所示。

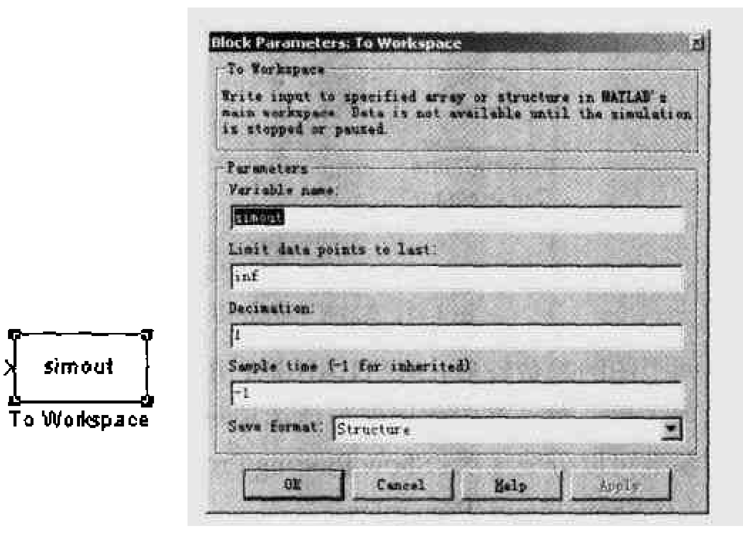



图 4-46　工作区写入模块及其参数设置

工作区写入模块主要有以下几个参数。

■　Variable name（变量名称）

工作区中用于保存数据的变量的名称。

■　Limit data points to last（保存数据的上界）

写入工作区的数据的最大个数，缺省值是 1000。

■　Decimation（抽取因子）

当抽取因子设置为 $n$ 时，工作区写入模块每隔 $n$ 个输入数据执行一次写入操作。抽取因子的缺省值是 1，表示把每个输入数据都保存到工作区中。

■ Sample time（抽样时间）

工作区写入模块的抽样间隔。

■ Save format（数据保存的格式）

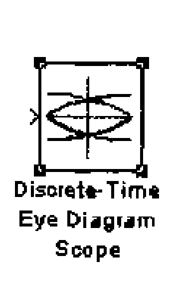
数据在工作区中保存的格式。数据保存格式有 3 种：数组格式（Array）、结构方式（Structure）以及带时间的结构方式（Structure with time）。

## 4.2.4 眼图、发散图和轨迹图

在通信系统的仿真过程中，眼图、发散图和轨迹图经常用于观测调制信号以及调制信号通过信道之后特征，以此来衡量调制方式和信道条件。本节我们将依次介绍离散时间眼图、离散时间发散图、离散时间轨迹图以及连续时间延图和发散图。

### 1. 离散时间眼图

离散时间眼图（Discrete Eye Diagram Scope）是用来产生眼图的模块，通常用于显示调制信号的输出。离散时间眼图模块只有一个输入端口，用于输入离散时间信号。这个信号既可以是实信号，也可以是复信号。离散时间眼图模块及其参数设置对话框如图 4-47 所示。



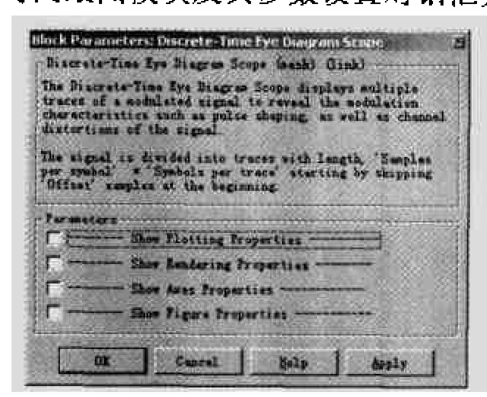




图 4-47　离散时间眼图模块及其参数设置

由于离散时间眼图的参数较多，MATLAB 把它的参数设置对话框分成几类。当选中了对话框中的某一选项之后，对话框中就出现与该选项对应的一类参数。从图 4-47 中可以看到，离散时间眼图的参数可以分成 4 类：Show Plotting Properties、Show Rendering Properties、Show Axes Properties 以及 Show Figure Properties。

Show Plotting Properties 参数用于确定眼图的绘制方式。当选中了“Show Plotting Properties”前面的复选框之后，参数设置对话框如图 4-48 所示。

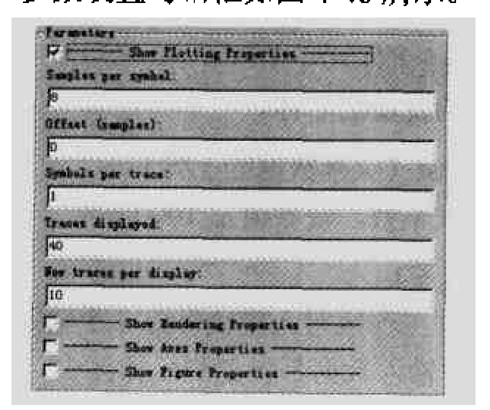



图 4-48　选中 Show Plotting Properties 选项之后的参数设置对话框

各参数的含义如下。

■　Samples per symbol（每个符号的抽样数）

指定在一个符号周期内的抽样次数。

■　Offset (samples)（抽样时延）

小于 Samples per symbol 的非负整数，表示 MATLAB 在开始绘制眼图之前应该忽略的抽样点的个数。

■　Symbols per trace（每径符号数）

对于每个输入信号，离散时间眼图模块可以同时绘制多条曲线，每条曲线称为一个“径”，它们在时间上相差一定的抽样周期。本参数指定了每一条径包含的输入符号的个数。假设每径符号数 Symbols per trace 为 $n$，每个符号的抽样数 Samples per symbol 等于 $m$，则每一条径的抽样点的个数等于 $m \times n$。

■　Traces displayed（显示径数）

离散时间眼图模块中显示的径的数目。本参数应该设置为一个正整数。

■　New traces per display（每次重新显示的径的数目）

小于 Traces displayed 的一个正整数，表示每次需要重新绘制的径的数目。

当选中了“Show Rendering Properties”前面的复选框之后，参数设置对话框如图 4-49 所示，里面列出了一些用于控制绘图属性的参数。




图 4-49　选择 Show Rendering Properties 选项之后的参数设置对话框

各参数的含义如下。

■　Markers（标识）

指定眼图中每个抽样点的绘制方式，如表 4-10 所示：当设置为字符“+”时，MATLAB 在每个抽样点上显示一个加号；当设置为字符“o”时，每个抽样点上显示为一个圆圈；当设置为字符“*”时，每个抽样点上显示为一个乘号；当设置为字符“.”时，每个抽样点上显示为一个点；当设置为字符“x”时，每个抽样点上显示为一个交叉。

表 4-10　　参数 Marker 的对应关系

| Marker | 说明 | 形状 |
|---|---|---|
| + | 每个抽样点上显示为一个加号 | +——————+——————+ |
| O | 每个抽样点上显示为一个圆圈 | o——————o——————o |

续表

| * | 每个抽样点上显示为一个乘号 | *————*————* |
|---|---|---|
| . | 每个抽样点上显示为一个点 | .————.————. |
| x | 每个抽样点上显示为一个交叉 | x————x————x |

■ Line style（线条样式）

眼图中线条绘制的样式。MATLAB 提供了 4 种样式的线条，如表 4-11 所示。

**表 4-11　线条绘制的样式**

| Line style | 形状 |
|---|---|
| Solid | ———————— |
| Dashed | - - - - - - - - |
| Dotted | ………………… |
| Dash-dot | -·-·-·-·-·- |

■ Line color（线条颜色）

眼图中线条的颜色。离散时间眼图能够绘制 5 种颜色的线条，如表 4-12 所示。

**表 4-12　线条绘制的颜色**

| Line color | RGB 值 |
|---|---|
| Black | (0,0,0) |
| Blue | (0,0,255) |
| Red | (255,0,0) |
| Green | (0,255,0) |
| Dark purple | (192,0,192) |

■ Duplicate points at trace boundary（迹边界上的重复点）

选中该选项前面的复选框之后，允许每条迹的边缘出现一个重复的抽样点；否则，不出现重复的点。

■ Color fading（颜色渐弱）

当选中该选项前面的复选框之后，眼图中每条迹上的点的深度随着仿真时间的推移而逐渐减弱。

■ High quality rendering（高质量绘制）

当选中该选项前面的复选框之后，离散时间眼图通过光栅操作绘制显示质量较高的眼图，这时候离散时间眼图的运行速度较慢；否则，离散时间眼图通过异或操作快速地绘制显示质量较低的眼图。

■ Show grid（显示网格）

本参数决定是否在眼图中显示网格线。

选择“Show Axes Properties”选项之后，参数设置对话框如图 4-50 所示，这时候可以按照自己的需要设置眼图中坐标轴的属性。

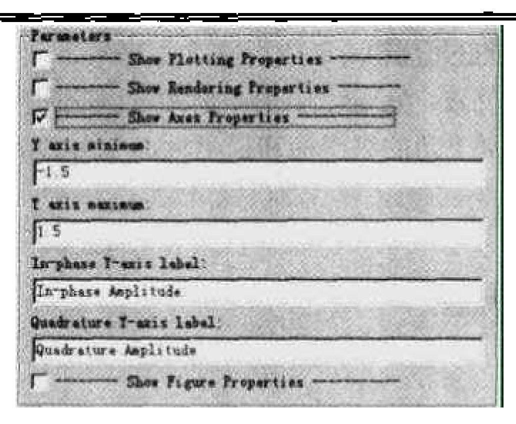



图 4-50　选择 Show Axes Properties 选项之后的参数设置对话框

- **Y-axis minimum（纵坐标最小值）**

本参数用于指定纵坐标的最小值，即输入信号强度的最小值。

- **Y-axis maximum（纵坐标最大值）**

本参数用于指定纵坐标的最大值，即输入信号强度的最大值。

- **In-phase Y-axis label（显示 I 支路纵坐标标签）**

本参数用于确定是否显示与 I 支路输入信号对应的纵坐标的标签。

- **Quadrature Y-axis label（显示 Q 支路纵坐标标签）**

本参数用于确定是否显示与 Q 支路输入信号对应的纵坐标的标签。

如果需要调整眼图的其他绘制属性，选中“Show Figure Properties”前面的复选框，这时候的参数设置对话框如图 4-51 所示。

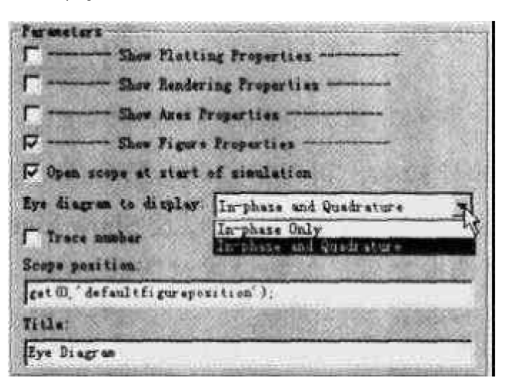



图 4-51　选择 Show Figure Properties 选项之后的参数设置对话框

- **Open scope at start of simulation（仿真开始时打开眼图）**

当选中了该选项前面的复选框之后，眼图将在仿真开始的时候自动打开；否则，用户需要双击离散时间眼图之后才能显示眼图。

- **Eye diagram to display（眼图显示的支路）**

本参数用于确定在眼图中显示哪个支路的输入信号：当选择“In-phase and Quadrature”时，MATLAB 在眼图中同时显示信号的 I 支路和 Q 支路；当选择“In-phase Only”时，MATLAB 只显示 I 支路信号，即只显示实信号。

- **Trace number（迹的编号）**

本参数用于确定是否在眼图中显示当前正在绘制的迹的编号。

■　Scope position（眼图的位置）

本参数用于确定眼图的位置，它是一个由 4 个元素组成的向量[left bottom width height]，left 和 bottom 分别表示眼图坐下角的纵坐标和横坐标，width 和 height 则分别表示眼图的宽度和高度。

■　Title（标题）

本参数用于指定眼图的标题。

### 2. 离散时间发散图

离散时间发散图（Discrete-Time Scatter Plot Scope）通常用来观测调制信号的特性和信道对调制信号的干扰特征。离散时间发散图模块接收复信号，并且根据输入信号绘制发散图。发散图的横坐标是输入复信号的 I 支路分量，纵坐标则是输入复信号的 Q 支路分量。离散时间发散图如图 4-52 所示。

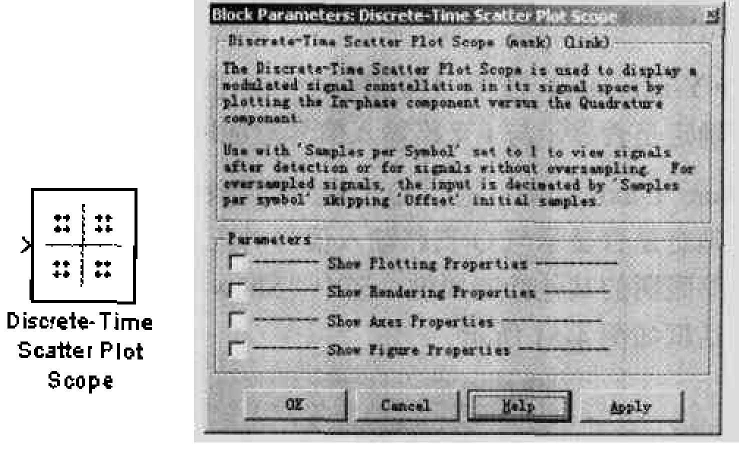



图 4-52　离散时间发散图模块及其参数设置

离散时间发散图的参数也比较多，可以把这些参数分成 4 类：Show Plotting Properties、Show Rendering Properties、Show Axes Properties 以及 Show Figure Properties。当选中“Show Plotting Properties”前面的复选框之后，参数设置对话框如图 4-53 所示。

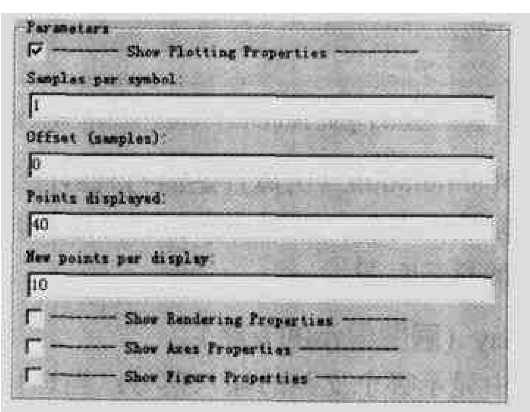



图 4-53　选择 Show Plotting Properties 选项之后的参数设置对话框

各参数的意义如下。

■　Samples per symbol（每个符号的抽样数）

本参数用来指定发散图中每个符号的抽样点的数目。

■　Offset (samples)（抽样时延）

小于 Samples per symbol 的非负整数，表示 MATLAB 在开始绘制发散图之前应该忽略的抽样点的个数。

■　Points displayed（点数）

离散时间发散图模块中显示的点的数目。

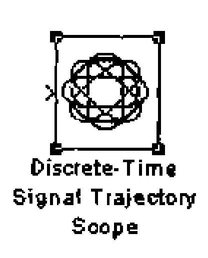
■　New points per display（每次重新显示的点的数目）

本参数用于确定离散时间发散图每次需要重新绘制的点的数目。

当选择 Show Rendering Properties，Show Axes Properties 以及 Show Figure Properties，相应的参数设置与 4.2.5 节中介绍的离散时间眼图相同。

### 3. 离散时间轨迹图

离散时间轨迹图（Discrete-Time Signal Trajectory Scope）根据输入的复信号绘制该信号的轨迹图。轨迹图的横坐标是输入复信号的 I 支路分量，纵坐标则是输入复信号的 Q 支路分量。离散时间轨迹图及其参数设置模块如图 4-54 所示。



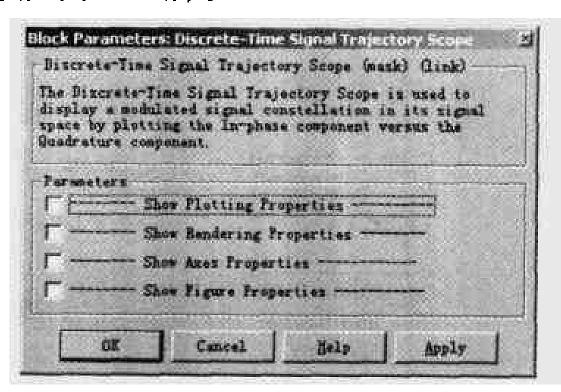




图 4-54　离散时间轨迹图模块及其参数设置

离散时间轨迹图的参数较多，从图 4-54 中可以看到，这些参数分成 4 类，分别为 Show Plotting Properties、Show Rendering Properties、Show Axes Properties 以及 Show Figure Properties。在参数设置对话框中选中“Show Plotting Properties”前面的复选框之后，对话框如图 4-55 所示。

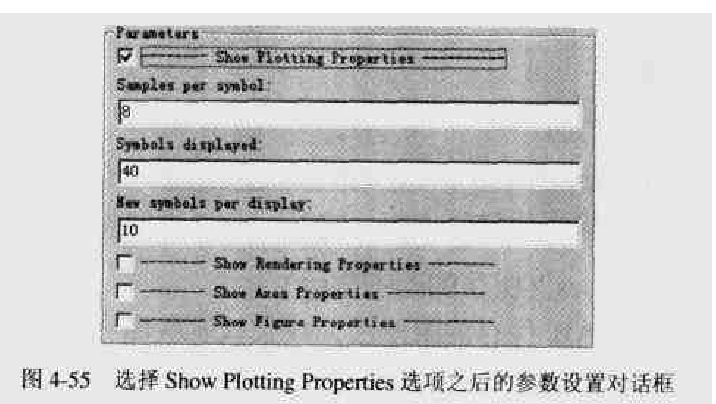



图 4-55　选择 Show Plotting Properties 选项之后的参数设置对话框

各参数含义如下。

■　Samples per symbol（每个符号的抽样数）

本参数用来指定发散图中每个符号的抽样点的数目。

■ Symbols displayed（符号数）

离散时间轨迹图中显示的符号的数目。

■ New points per display（每次重新显示的点的数目）

本参数用于确定离散时间轨迹图每次需要重新绘制的点的数目。

当选择“Show Rendering Properties”、“Show Axes Properties”以及“Show Figure Properties”时，相应的参数设置与 4.2.5 节中介绍的离散时间眼图相同。

### 4. 连续时间眼图和发散图

连续时间眼图和发散图（Continuous-Time Eye and Scatter Diagrams）用于绘制连续时间信号的眼图、发散图或 X-Y 坐标图。连续时间眼图和发散图模块及其参数设置对话框如图 4-56 所示。

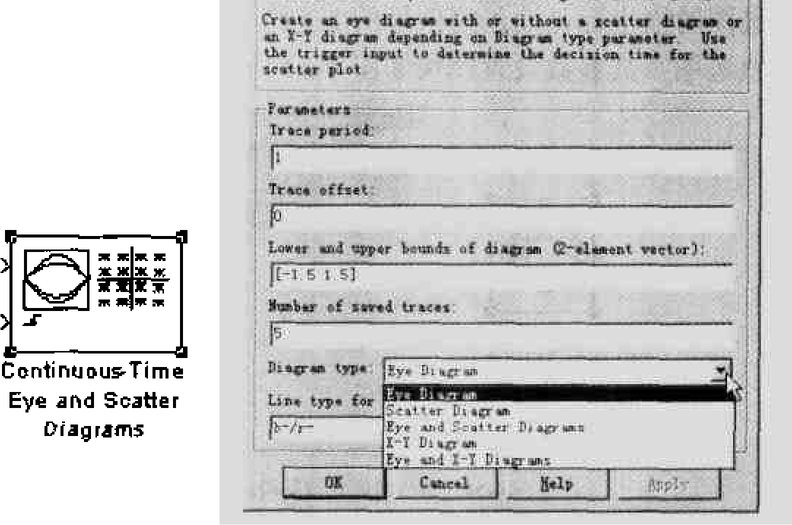



图 4-56 连续时间眼图和发散图模块及其参数设置

连续时间眼图和发散图主要有以下几个参数。

■ Trace period（周期）

连续时间眼图的绘制周期，对应于眼图中的横坐标的尺度大小。

■ Trace offset（偏移）

眼图横坐标中左端对应的时间点。

■ Lower and upper bounds of diagram（眼图的上界和下界）

本参数用于设置信号的最大值和最小值，它是一个二维向量。

■ Number of saved traces（保存的迹的数目）

本参数用于指定 MATLAB 保存的眼图的曲线数或发散图中点的数目。当眼图或发散图重新设置大小之后，MATLAB 根据保存的数据重新绘制图形，从而恢复出指定数目的迹。

■ Diagram type（图的类型）

本参数用于设置显示的图的类型。当设置为 Eye Diagram 时，只绘制眼图；当设置为 Scatter Diagram 时，只绘制发散图；当设置为 Eye and Scatter Diagrams 时，同时绘制眼图和发散图；

当设置为 X-Y Diagram 时，绘制 X-Y 图；当设置为 Eye and X-Y Diagrams 时，同时绘制眼图和 X-Y 图。

■ Line type for eye diagram（眼图的线条设置）

当 Diagram type 包含了眼图时本参数有效，本参数用于指定眼图中线条绘制的形状和颜色。线条设置的参数如表 4-13 所示。在设置参数的时候，依次从表 4-13 中选择与所需的颜色、标识和线条类型对应的参数，如 gd:表示绿色点线，其中每个抽样点表示成一个菱形。如果需要同时设置复信号的实部和虚部的线条属性，则在实部和虚部的线条属性之间加上一个斜杠“/”，如 gd:/ro-表示实部的线条设置是 gd:，虚部的线条设置是 ro-。

**表 4-13　　线条设置参数**

| 颜色设置 | | 标识类型 | | 线条类型 | |
|---|---|---|---|---|---|
| Y | 黄色 | . | 点 | - | 实线 |
| M | 紫红色 | o | 阆圈 | : | 点线 |
| C | 青色 | x | 叉号 | -. | 点划线 |
| R | 红色 | + | 加号 | -- | 划线 |
| G | 绿色 | * | 星号 | | |
| B | 蓝色 | s | 方框 | | |
| W | 白色 | d | 菱形 | | |
| K | 黑色 | v | 向下三角形 | | |
| | | ^ | 向上三角形 | | |
| | | < | 向左三角形 | | |
| | | > | 向右三角形 | | |
| | | p | 五角星 | | |
| | | h | 六角星 | | |

■ Line type for scatter diagram（发散图的线条设置）

当 Diagram type 包含了发散图时本参数有效，用于指定眼图中线条绘制的标记和颜色。

■ Line type for X-Y diagram（X-Y 图的线条设置）

当 Diagram type 包含了 X-Y 图时本参数有效，用于指定眼图中线条绘制的形状和颜色。

# 第 5 章 信道

信道是通信系统的基本环节之一。在通信系统中，发送端产生的数据通过信源编码和信号调制转化成调制信号，然后进入信道。这些调制信号通过信道到达接收端，在接收端通过与发送端相反的过程得到原始数据。信道的传输质量影响着信号的接收和解调。这种影响表现在两个方面，一是产生噪声，二是减弱信号的强度和改变信号的形状。通信系统的设计目标是使接收端能够毫无差别地再现发送端发送的原始数据。因此，在设计通信系统的过程中，信道对传输信号的影响是一个不可或缺的环节。

在信号传输的过程中，它会不可避免地受到各种干扰，这些干扰统称为“噪声”。加性高斯白噪声 AWGN（Additive White Gaussian Noise）是最常见的一种噪声，它存在于各种传输媒质中，包括有线传输信道和无线传输信道。在无线信道中，信号在受到加性高斯白噪声干扰的同时，还会受到瑞利衰落或伦琴衰落的影响。这种影响表现为信号的一种快速衰落过程，它对无线信号的传输质量起着决定性的作用，因此，无线通信系统的很多研究工作都是围绕着如何降低这种干扰进行的。

根据信道中占据主导地位的噪声的特点，信道可以分成加性高斯白噪声信道、二进制对称信道、多径瑞利衰落信道和伦琴衰落信道等。本节将通过具体的实例依次介绍这 4 种信道的原理和实现。

值得注意的是，由于噪声表现为一种随机过程，因此关于噪声的各种参数，如均值和方差（即噪声信号的幅度和功率），都是一种统计平均值，是以大量的观测样本为基础计算得到的统计数值。

## 5.1 加性高斯白噪声信道

加性高斯白噪声是最简单的一种噪声，它表现为信号围绕平均值的一种随机波动过程。加性高斯白噪声的均值为 0，方差表现为噪声功率的大小。一般情况下，噪声功率越大，信号的波动幅度就越大，接收端接收到的信号的误比特率就越高。在研究通信系统的误码率与信道质量的关系时，一般先研究它在加性高斯白噪声信道下的性能，然后再把它推广到具有快衰落的复杂情况。

### 5.1.1 函数 awgn()

函数 awgn()在输入信号中叠加一定强度的高斯白噪声信号，噪声信号的强度由函数的参数确定。函数 awgn()有以下几种形式：

（1）y = awgn(x,snr)

函数 awgn()把加性高斯白噪声叠加到输入信号 x 中，噪声的强度由信噪比 snr（单位：dB）确定，并且信号 x 的强度假定为 0 dBW。此时，噪声信号的功率实际上就等于-snr dBW。下面的代码段产生一个强度为 2 的信号，然后叠加一个信噪比等于 10dB 的高斯白噪声，并且计算这个信号的均值和方差。

```
>> x(1:10^6) = 2;
>> y = awgn(x, 10);
>> mean(y)
ans =
     2.0001
>> var(y)
ans =
     0.1000
```

从这个例子中可以清楚地看出，加性高斯白噪声服从均值为 x = 2，方差为 0.1 的高斯分布，而且这个方差是由 snr 决定的。

（2）y = awgn(x,snr,sigpower)

指定了输入信号的功率为 sigpower（单位：dBW）。下面的代码段在强度为 2 的信号上叠加高斯白噪声，并且设定输入信号的功率为 10dBW。

```
>> x(1:10^6) = 2;
>> y = awgn(x, 10, 10);
>> mean(y)
ans =
     1.9999
>> var(y)
ans =
     1.0001
```

（3）y = awgn(x,snr,'measured')

输入信号的功率根据输入信号 $x$ 的数值进行计算。如果 $x$ 是向量，则在计算 $y(i)$的时候，MATLAB 首先计算 $x(i)$的功率 $P(i) = x(i)^2$，通过 $P(i)$和 $snr$ 计算出方差 var($i$)，最后以 $x(i)$为均值，var($i$)为方差计算高斯随机过程。下面的代码段产生一个高斯白噪声，并且高斯白噪声的强度取决于输入信号的强度。

```
>> x = 1:10;
>> y = awgn(x, 10, 'measured')
y =
  Columns 1 through 8
    4.4321    2.2602    3.6381    2.4424    5.6179    4.9654    8.8292   10.2852
  Columns 9 through 10
    4.9861    8.7356
```

从上面的例子中可以看出，$y(i)$的方差随着 $x(i)$的变化而发生相应的变化。

（4）y = awgn(x,snr,sigpower,state)

MATLAB 计算加性高斯白噪声前，将随机数 randint 重新设置为 state。通过这种方式，MATLAB 每次调用相同的语句的时候都将得到相同的结果。下面的代码段用随机数 3 对内部状态实施初始化，然后产生相应的高斯白噪声。

```
>> y = awgn(2, 10,'measured',3)
y =
    2.5869
>> y = awgn(2, 10,'measured',4)
y =
    3.1451
>> y = awgn(2, 10,'measured',3)
y =
    2.5869
```

从上面的例子中可以看出，当使用相同的随机数 3 时，得到相同的输出 $y$ = 2.5869，而使用另外一个随机数 4 得到的结果却是 $y$ = 3.1451。

（5）y = awgn(...,powertype)

powertype 指明了 snr 和 sigpower 的单位，这些单位可以是 db，也可以是 linear。当使用 db 为单位时，snr 的单位是 dB，sigpower 的单位是 dBW。如果 powertype 设置为 linear， snr 使用绝对值，sigpower 的单位是瓦（watts）。

## 5.1.2 函数 wgn()

函数 wgn()产生高斯白噪声（White Gaussian Noise）。通过 wgn()函数可以产生实数形式或复数形式的噪声，噪声的单位可以是 dBW、dBm 或绝对数值。该函数有以下几种形式：

（1）y = wgn(m,n,p)

产生 $m$ 行 $n$ 列的白噪声矩阵，$p$ 表示输出信号 $y$ 的功率（单位：dBW），并且设定负载的电阻为 1 欧姆。下面的代码段产生功率为 10dBW 的高斯白噪声。

```
>> y = wgn(1,10^6,10);
>> mean(y)
ans =
    0.0029
>> var(y)
ans =
    10.0251
```

在这个例子中，输出噪声信号的功率设置为 10dBW（即 10W），而 y 的方差（10.0251）也说明了这一点。

（2）y = wgn(m,n,p,imp)

指定负载的电阻为 imp（单位：欧姆）。下面的代码段产生强度为 10dBW、负载为 2 欧

姆的高斯白噪声。

```
>> y = wgn(1, 10^6, 10, 2);
>> mean(y)
ans =
     0.0041
>> var(y)
ans =
    19.9938
```

设定负载的电阻为 2 欧姆，输出噪声信号的功率依然是 10dBW，则由 $y^2 = P \times R$ 得到 $y^2 = 20 \times V^2$。由于 $y$ 的均值为 0，因此 var($y$)应该等于 20，这与示例程序的结果相吻合。

（3） y = wgn(m,n,p,imp,state)

在计算加性高斯白噪声前将随机数 randint 重新设置为 state。通过这种方式，MATLAB 每次调用相同 wgn()的语句的时候都将得到相同的结果。下面的代码段使用随机数 5 对内部状态进行初始化，然后产生强度为 10dBW、负载为 2 欧姆的高斯白噪声。

```
>> y = wgn(1, 5, 10, 2, 5)
y =
     4.4169     1.1585    -1.7000      0.7720     -6.8379
>> y = wgn(1, 5, 10, 2, 6)
y =
     8.3883     1.3600     4.8355     -1.1455      6.2201
>> y = wgn(1, 5, 10, 2, 5)
y =
     4.4169     1.1585    -1.7000      0.7720     -6.8379
```

当使用相同的随机数时（state = 5），输出信号是相同的。而当时用不同的 state 时，输出噪声信号并不相等

（4） y = wgn(...,powertype)

参数 powertype 指明了输出信号功率 $p$ 的单位，这些单位可以是 dBW、dBm 或 linear。下面的代码段使用随机数 1 产生强度为 1W、负载为 1 欧姆的高斯白噪声。

```
>> y = wgn(1, 5, 1, 1, 1, 'linear')
y =
     0.8644     0.0942    -0.8519      0.8735    -0.4380
>> y = wgn(1, 5, 0, 1, 1, 'dBW')
y =
     0.8644     0.0942    -0.8519      0.8735    -0.4380
>> y = wgn(1, 5, 30, 1, 1, 'dBm')
y =
     0.8644     0.0942    -0.8519      0.8735    -0.4380
```

上面的例子分别使用了不同的功率单位，由于 1 W = 0 dBW = 30 dBm，并且使用了相同的随机数，因此产生的输出信号是相同的。

（5）y = wgn(...,outputtype)

参数 outputtype 用于指定输出信号的类型。当 outputtype 被设置为 real 时输出实信号。当 outputtype 被设置为 complex 时，输出信号的实部和虚部的功率都为 $p/2$。下面的代码段使用随机数 1 产生强度为 10dBW、负载为 1 欧姆的复数形式的高斯白噪声。

```
>> y = wgn(1, 10^6, 10, 1, 1, 'dBW', 'complex');
>> mean(y)
ans =
   0.0019 + 0.0005i
>> var(y)
ans =
   10.0123
>> var(real(y))
ans =
    4.9979
>> var(imag(y))
ans =
     5.0145
```

在上面的例子中，输出信号的总功率是 10 dBW = 10 W，而实部和虚部的功率都约等于 5 W。

## 5.1.3 加性高斯白噪声信道模块

加性高斯白噪声信道（AWGN Channel）模块位于 Blockets | Communications Blockset | Channels | AWGN Channel，它的作用是在输入信号中加入高斯白噪声。图 5-1 所示是加性高斯白噪声信道模块及其属性设置对话框。

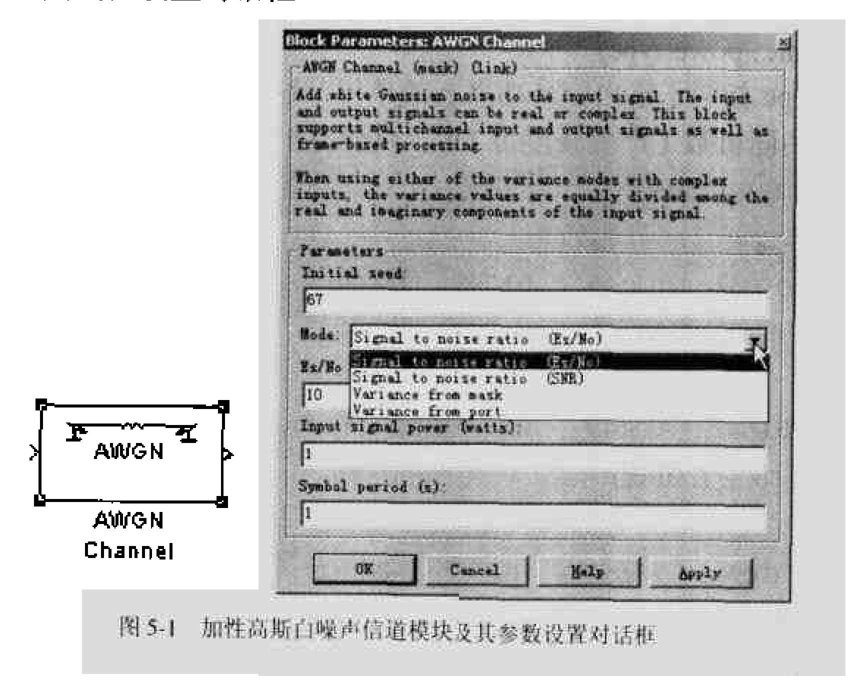



图 5-1　加性高斯白噪声信道模块及其参数设置对话框

加性高斯白噪声信道模块有一个输入端口和一个输出端口，输入信号既可以是实信号，也可以是复信号。当操作模式设置为 Variance from port 时，加性高斯白噪声信道模块具有两个输入端口，其中第二个输入端口输入的是表示高斯白噪声方差的信号。加性高斯白噪声信道模块有以下几个参数。

■ Initial seed（初始化种子）

加性高斯白噪声信道模块的初始化种子。不同的 Initial seed 对应于不同的输出，相同的 Initial seed 产生相同的输出。因此，只要设置相同的 Initial seed 就能够再现相同的随机过程，这种特性对于需要多次重复的仿真过程来说是相当重要的。当输入信号是一个 $m\times n$ 的矩阵时，Initial seed 可以是一个 $n$ 维向量，向量中的每个元素对应一列输入信号。

■ Mode（操作模式）

加性高斯白噪声信道模块的操作模式。当 Mode 设置为 Signal to noise ratio ($E_s/N_0$)时，加性高斯白噪声信道模块根据信噪比 $E_s/N_0$ 确定高斯白噪声的功率，这时候需要确定 3 个参数：信噪比 $E_s/N_0$、输入信号功率以及信号周期。当 Mode 设置为 Signal to noise ratio (SNR)时，加性高斯白噪声信道模块根据信噪比 SNR 确定高斯白噪声的功率，这时候需要确定两个参数：信噪比 SNR 以及周期。当 Mode 设置为 Variance from mask 时，加性高斯白噪声信道模块根据方差确定高斯白噪声的功率，这个方差由 Variance 指定，并且必须是正数。当 Mode 设置为 Variance from port 时，加性高斯白噪声信道模块有两个输入：一个用于输入信号，另外一个输入用于确定高斯白噪声的方差。

对于复数形式的输入信号，加性高斯白噪声信道模块中的 $E_s/N_0$ 和 SNR 具有公式 5.1 所示的关系：

$$E_s / N_0 = SNR\cdot(T_{sym} / T_{samp}) \tag{5.1}$$

其中，$T_{sym}$ 表示输入信号的符号周期，$T_{samp}$ 表示输入信号的抽样周期。由于在加性高斯白噪声信道模块中复信号的噪声功率谱密度等于 $N_0$，而实信号的噪声功率谱密度等于 $N_0/2$，因此对于实数形式的输入信号，$E_s/N_0$ 和 SNR 之间的关系可以表示成公式 5.2 所示：

$$E_s / N_0 = 2\cdot SNR\cdot(T_{sym} / T_{samp}) \tag{5.2}$$

■ $E_s/N_0$ (dB)（信噪比）

加性高斯白噪声信道模块的信噪比 $E_s/N_0$（单位：dB）。本参数在 Mode 属性设置为 Signal to noise ratio ($E_s/N_0$)时有效。

■ SNR (dB)（信噪比）

加性高斯白噪声信道模块的信噪比 SNR（单位：dB）。本参数在 Mode 属性设置为 Signal to noise ratio (SNR)时有效。

■ Input signal power (watts)（输入信号功率）

加性高斯白噪声信道模块输入信号的平均功率（单位：瓦特）。本参数在 Mode 属性设置为 Signal to noise ratio ($E_s/N_0$)或 Signal to noise ratio (SNR)时有效。当 Mode 属性等于 Signal to noise ratio ($E_s/N_0$)时，Input signal power 表示输入符号的均方根功率；当 Mode 属性等于 Signal to noise ratio (SNR)时，Input signal power 表示输入的抽样信号的均方根功率。

■ Symbol period (s)（符号周期）

加性高斯白噪声信道模块每个输入符号的周期（单位：秒）。本参数在 Mode 属性设置为 Signal to noise ratio ($E_s/N_0$)时有效。

■ Variance（方差）

加性高斯白噪声信道模块产生的高斯白噪声信号的方差。本参数在 Mode 属性设置为 Variance from mask 时有效。

### 5.1.4 实例 5.1——BFSK 在高斯白噪声信道中的传输性能

AMPS（Advanced Mobile Phone System）是由 AT&T Bell 实验室开发的最早的移动通信系统，它采用二进制频移键控（BFSK）对信号进行调制。在 AMPS 中，信道的带宽是 30 kHz，话音信道的频率间隔是 24 kHz，控制信道的频率间隔是 16 kHz，信道的传输速率为 10 kbit/s。在本小节中，我们将结合 BFSK 介绍高斯白噪声信道（AWGN Channel）模块的一个应用。

信噪比决定了信号传输的质量，信噪比与信道的误码率一般是成反比的。实例 5.1 将探讨二进制频移键控在高斯白噪声信道中的传输。图 5-2 所示是该实例的系统结构框图。从图中可以看到，实例 5.1 由 3 个部分组成：Source（信源模块）、Channel（信道模块）以及 Sink（信宿模块）。

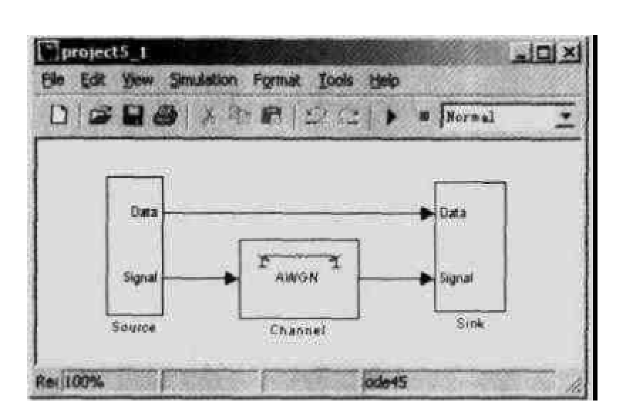



图 5-2 实例 5.1 系统框图

在本实例中，Source（信源模块）产生速率为 10 kbit/s、帧长度为 1 秒的二进制数据 Data，并且通过 BFSK 调制产生调制信号 Signal。调制信号 Signal 通过高斯白噪声 Channel（信道模块）后成为 Signal Out，信号的信噪比等于 SNR。Sink（信宿模块）对 Signal 进行解调，并且把解调后的数据与 Source（信源模块）产生的原始数据进行比较，根据比较的结果计算误比特率。最后，Sink（信宿模块）根据信噪比 SNR 与误比特率的对应关系绘制对数曲线图。下面分别介绍各个模块。

■ Source（信源模块）

信源模块（Source）产生数据的速率为 10 kbit/s，每帧的周期为 1 秒，因此，一帧长度等于 10000 bit。图 5-3 所示是 Source（信源模块）的结构框图，它由两部分组成：Random Integer Generator（随机整数产生器）和 M-FSK Modulator Baseband（BFSK 基带调制器）。

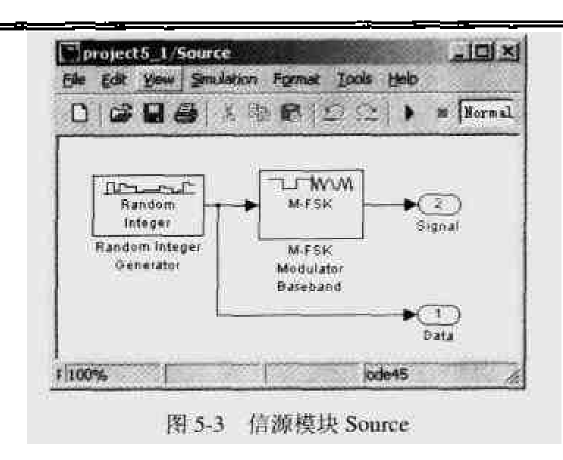



图5-3　信源模块 Source

随机整数产生器产生的数据一方面作为输出信号 Data，另一方面进入 BFSK 基带调制器模块，由 BFSK 基带调制器对数据进行 BFSK 调制，输出调制信号 Signal。随机整数产生器和 BFSK 基带调制器的参数设置如表 5-1 和表 5-2 所示。

**表5-1　Random Integer Generator（随机整数产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Random Integer Generator |
| M-ary number | 2 |
| Initial seed | 37 |
| Sample time | 1/BitRate |
| Frame-based outputs | Checked |
| Samples per frame | BitRate |

**表5-2　M-FSK Modulator Baseband（BFSK 基带调制器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | M-FSK Modulator Baseband |
| M-ary number | 2 |
| Input type | Bit |
| Symbol set ordering | Binary |
| Frequency separation (Hz) | FrequencySeparation |
| Phase continuity | Continuous |
| Samples per symbol | SamplesPerSymbol |

■　Sink（信宿模块）

在信宿模块（Sink）中，M-FSK Demodulator Baseband（BFSK 基带解调器）对接收信号进行解调，然后通过误码率计算器（Error Rate Calculation）计算该帧的误比特率。误码率计算器产生的数据是一个三维向量，分别表示误码率、误码个数以及信号总数，因此，通过一个 Selector（选择器）选择向量的第一个元素作为输出信号。这个输出信号进入 To Workspace（工作区）模块，并且保存为变量 BitErrorRate。信宿模块的结构框图如图 5-4 所示。

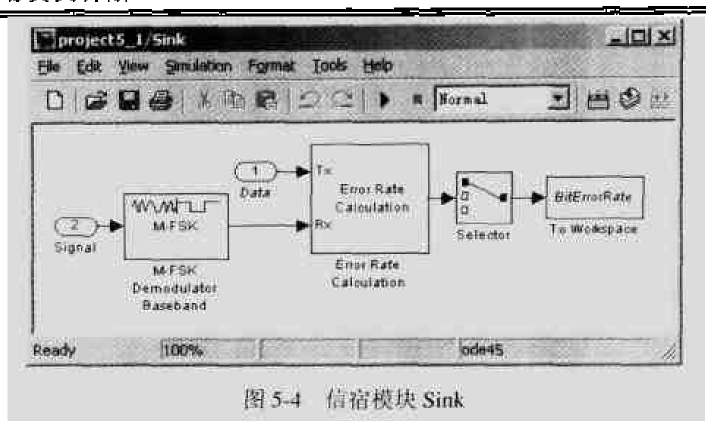



图 5-4 信宿模块 Sink

BFSK 基带解调器、误码率计算器、选择器和工作区写入模块的参数设置分别如表 5-3~表 5-6 所示。

表5-3 M-FSK Demodulator Baseband（BFSK 基带解调器）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | M-FSK Demodulator Baseband |
| M-ary number | 2 |
| Output type | Bit |
| Symbol set ordering | Binary |
| Frequency separation (Hz) | FrequencySeparation |
| Samples per symbol | SamplesPerSymbol |

表5-4 Error Rate Calculation（误码率计算器）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Error Rate Calculation |
| Receive delay | 0 |
| Computation delay | 0 |
| Computation mode | Entire frame |
| Output data | Port |
| Reset port | Checked |
| Stop simulation | Unchecked |

表5-5 Selector（选择器模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Selector |
| Input Type | Vector |
| Source of element indices | Internal |
| Elements | 1 |
| Input port width | 3 |

表5-6　　To Workspace（工作区写入模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | To Workspace |
| Variable name | BitErrorRate |
| Limit data points to last | inf |
| Decimation | 1 |
| Sample time | -1 |
| Save format | Array |

■　Channel（信道模块）

最后一个模块是我们这个实例的核心模块：Channel（信道模块）。在本实例中，Channel（信道模块）就是一个 AWGN Channel（加性高斯白噪声产生器），它将噪声叠加到信源模块产生的 BFSK 调制信号中。AWGN Channel（加性高斯白噪声噪声产生器）的参数设置如表 5-7 所示。

表5-7　　加性高斯白噪声产生器（AWGN Channel）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | AWGN Channel |
| Initial Seed | 67 |
| Mode | Signal to noise ratio (SNR) |
| SNR (dB) | SNR |
| Input signal power (watts) | 1 |

最后，将运行参数 Simulation | Simulation Parameters | Stop Time 设置为 SimulationTime。在我们这个实例中，所有以字符串命名的参数（如 SimulationTime、FrequencySeparation 等）都应该在工作区中创建相应的变量。在这里我们将在脚本程序中对这些字符串的数值进行定义。

本实例程序需要运行多次才能够得到信道的信噪比与信号的误比特率之间的关系，为此需编写如下的脚本程序（保存为文件 project5_1main.m）：

```
% x 表示信噪比
x=0:15;
% y 表示信号的误比特率，它的长度与 x 相同
y=x;

% BFSK 调制的频率间隔等于 24kHz
FrequencySeparation=24000;
% 信源产生信号的 bit 率等于 10kbit/s
BitRate=10000;
%
```

```
% BFSK 调制信号每个符号的抽样数等于 2
SamplesPerSymbol=2;

% 循环执行仿真程序
for i=1:length(x)
    % 信道的信噪比依次取 x 中的元素
    SNR=x(i);
    % 运行仿真程序，得到的误比特率保存在工作区变量 BitErrorRate 中
    sim('project5_1');
    % 计算 BitErrorRate 的均值作为本次仿真的误比特率
    y(i)=mean(BitErrorRate);
end

% 准备一个空白的图
hold off;
% 绘制 x 和 y 的关系曲线图，纵坐标采用对数坐标
semilogy(x,y);
```

在 MATLAB 工作区中输入命令行“project5_1main”（注意把工作区的当前路径设置为文件 project5_1main.m 所在的目录），程序运行结束之后得到一个如图 5-5 所示的曲线图。在这个曲线图中，$X$ 轴表示的是信噪比 SNR（单位：dB），$Y$ 轴表示的是信号的误比特率（对数坐标）。从图中可以看出，在加性高斯白噪声（AWGN）信道中，BFSK 调制信号的误比特率随着信噪比的增加而降低，当信噪比达到 14 dB 时，误比特率低于$10^{-3}$。

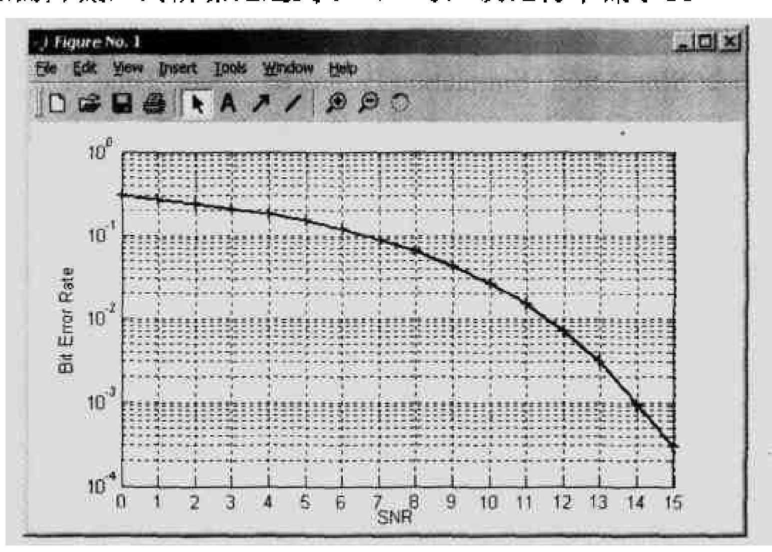



图 5-5　误比特率与信噪比的关系曲线图

事实上，当采用相干检测时，BFSK 调制的误比特率与信号的信噪比之间有公式 5.3 所示关系：

$$P_{e,BFSK}=Q\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{5.3}$$

本实例中的误码率明显高于公式 5.3 的计算结果。出现这种结果的原因在于，在仿真过程中，BFSK 调制性能的好坏除了跟信号的信噪比有关之外，还与抽样速率（即 M-FSK Modulator Baseband 和 M-FSK Demodulator Baseband 中的 Samples per symbol 参数）和仿真时间（Simulation time）有关。在这个例子中，我们把 Samples per symbol 设置为 2，即每个信号仅抽样两次。当把 Samples per symbol 设置为更大的数值时，仿真结果中得到的误码率将大大降低，并且逐渐趋向于根据上面的公式计算得到的数值。需要注意的是，当增大 Samples per symbol 的数值时，仿真时间 Simulation time 的数值也要相应地增加，这时候仿真所需的时间也将会延长。

# 5.2　二进制对称信道

二进制对称信道一般用于对二进制信号的误比特率性能进行仿真。二进制对称信道产生一个二进制噪声序列，在这个序列中，“1”出现的概率就是二进制对称信道的误码率。输入的二进制信号序列与这个二进制噪声序列异或之后，就得到二进制对称信道的输出信号。

## 5.2.1　二进制对称信道模块

二进制对称信道模块（Binary Symmetric Channel）的输入和输出都是二进制信号，它在输入信号中加入二进制噪声信号。二进制噪声信号是由“0”和“1”组成的信号，其中“0”表示没有传输错误，“1”表示产生了传输错误，“1”出现的概率就是二进制对称信道的误比特率。图 5-6 所示是二进制对称信道模块及其参数设置对话框。

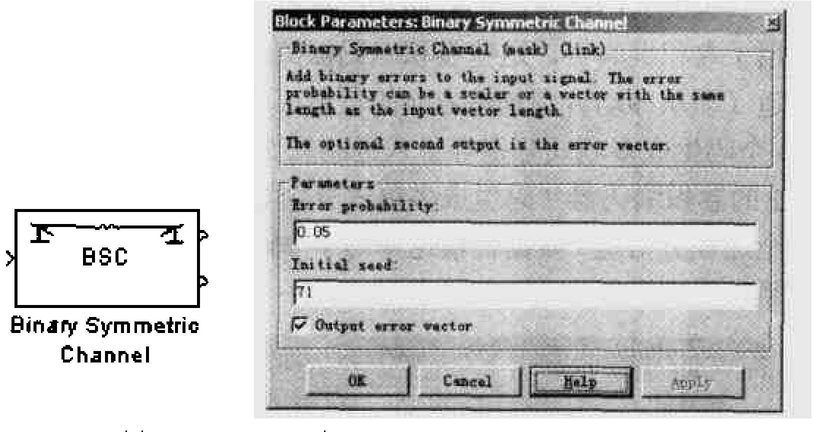



图 5-6　二进制对称信道模块及其参数设置对话框

二进制对称信道模块有以下 3 个参数：

■　Error probability（误比特率）

二进制对称信道模块的误比特率。本参数是一个介于 0 和 1 之间的实数。

■　Initial seed（初始化种子）

二进制对称信道模块的初始化种子。不同的 Initial seed 对应于不同的输出，相同的 Initial seed 产生相同的输出。

■ Output error vector（误差输出）

当选择本参数前面的复选框之后，二进制对称信道模块有两个输出端口，其中第二个输出端口输出的是二进制噪声信号。当取消选择本参数后，二进制对称信道模块只有一个输出端口，这时候只输出加入了二进制噪声后的信号。

二进制对称信道模块是由一系列简单模块构造而成的，如图 5-7 所示。在二进制对称信道模块中，Random Source（随机数发生器）产生一个在 0 和 1 之间的实数，并且把这个随机数与常数 P（P 对应于二进制对称信道模块的误比特率）进行比较。如果产生的随机数小于 P，则产生的二进制噪声信号等于 1，这个噪声信号与输入信号执行异或操作之后将改变输入信号的数值，从而产生一个传输错误。

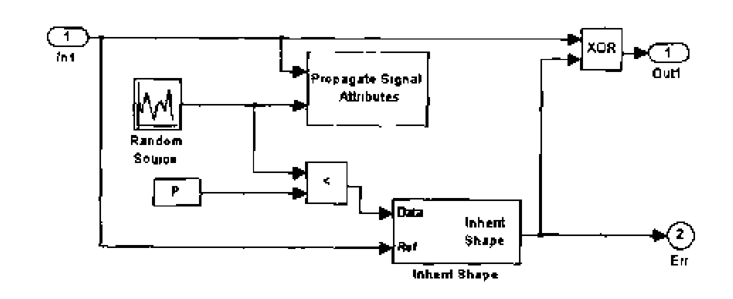



图 5-7　二进制对称信道模块的内部结构

## 5.2.2　实例 5.2——卷积编码器在二进制对称信道中的性能

二进制对称信道模块的一个主要用途是用于检验编码的纠错和检错性能，因为它能够方便地建立信道误比特率与编码信号的误比特率之间的关系。本节以卷积编码为例介绍卷积编码器在二进制信道中的传输性能。

卷积编码器以其优良的纠错性能在移动通信系统中得到了广泛的应用。IS-95 的前向信道和反向信道都采用了卷积编码器，这些卷积编码器的约束长度都是 9，码率等于 1/2 或 1/3。对于反向业务信道（即从移动台到基站方向的信道），速率集 1（Rate Set 1）采用码率为 1/3 的卷积编码器（3 个码生成多项式分别等于八进制数 557、663 和 711），速率集 2（Rate Set 2）则采用码率为 1/2 的卷积编码器（码生成多项式分别等于八进制数 753 和 561）。在本实例中我们将使用反向全速业务信道（数据传输速率等于 9600bit/s）速率集 1 的卷积编码器。图 5-8 所示是本实例的系统组成框图。

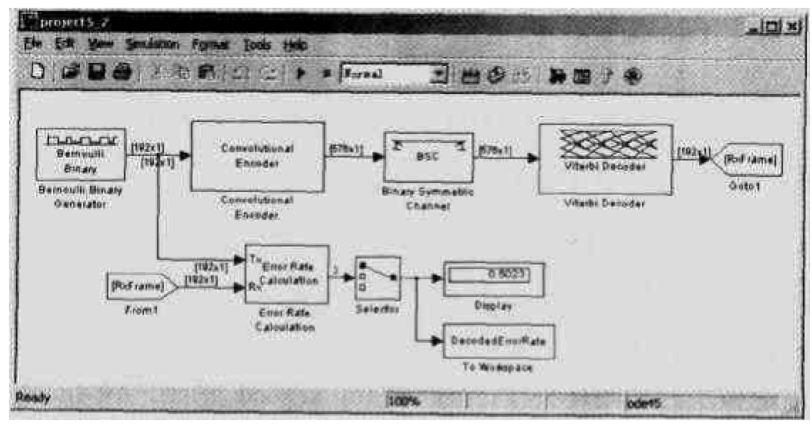


图 5-8　实例 5.2 的系统组成框图

在本实例中，我们使用一个 Bernoulli Binary Generator（二进制贝努利序列生成器）作为数据源，产生一个长度为 192 bit 的二进制数据帧，它对应于 9600 bit/s 业务信道的一个 20 毫秒数据帧。二进制贝努利序列生成器的参数设置如表 5-8 所示。

**表5-8　Bernoulli Binary Generator（二进制贝努利序列生成器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | 61 |
| Sample time | 1/9600 |
| Frame-based outputs | Checked |
| Samples per frame | 9600 |

二进制贝努利序列生成器产生的 20 毫秒数据帧通过码率为 1/3 的 Convolutional Encoder（卷积编码器）进行编码，编码后得到长度为 576 bit 的二进制数据帧。编码后的数据帧通过一个 Binary Symmetric Channel（二进制对称信道），由于噪声的干扰，其中的某些数据位将改变数值。这些数据帧最后进入 Viterbi Decoder（维特比译码器），由译码器对一帧数据进行纠错，得到即标号为 RxFrame 的信号（译码数据帧）。卷积编码器、二进制对称信道和维特比译码器的参数设置分别如表 5-9~表 5-11 所示。

**表5-9　Convolutional Encoder（卷积编码器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Convolutional Encoder |
| Trellis structure | poly2trellis(9,[557,663,711]) |
| Reset | On each frame |

**表5-10　Binary Symmetric Channel（二进制对称信道）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Binary Symmetric Channel |
| Error probability | BitErrorRate |
| Initial seed | 71 |
| Output error vector | Unchecked |

**表5-11　Viterbi Decoder（维特比译码器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Viterbi Decoder |
| Trellis structure | poly2trellis(9,[557,663,711]) |
| Decision type | Hard Decision |
| Traceback depth | 192 |
| Operation mode | Truncated |

为了统计解码后数据的误比特率，数据源产生的二进制数据帧和维特比译码器产生的译码数据帧分别作为发送端数据和接收端数据进入 Error Rate Calculation（误码率统计模块）。

误码率统计模块的输出信号是一个长度为 3 的向量，其中第一个元素表示误比特率，第二个元素表示误码的个数，第三个元素则表示输入信号的个数。通过一个 Selector（选择器），我们把误码率统计模块输出向量的第一个元素（即误比特率）输出到 Display（显示模块）中。同时为了计算误比特率的平均值，我们还通过 To Workspace（工作区写入模块）把误比特率输出到工作区，并且保存在变量 DecodedErrorRate 中。表 5-12~表 5-15 所示分别是误码率统计模块、选择器模块、显示模块以及工作区写入模块的参数设置。

**表5-12　Error Rate Calculation（误码率统计模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Error Rate Calculation |
| Receive delay | 0 |
| Computation delay | 0 |
| Computation mode | Entire frame |
| Output data | Port |
| Reset port | Unchecked |
| Stop simulation | Unchecked |

**表5-13　Selector（选择器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Selector |
| Input Type | Vector |
| Source of element indices | Internal |
| Elements | 1 |
| Input port width | 3 |

**表5-14　Display（显示模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Display |
| Format | Short |
| Decimation | 1 |
| Floating display | Unchecked |
| Sample time | -1 |

**表5-15　To Workspace（工作区写入模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | To Workspace |
| Variable name | DecodedErrorRate |
| Limit data points to last | inf |
| Decimation | 1 |
| Sample time | -1 |
| Save format | Array |

到这里为止，整个系统已经建造完毕。要运行实例程序，首先在工作区中建立一个变量 BitErrorRate，并且把它设置为所需的二进制对称信道的误比特率。在这个程序里我们使用了缺省的运行时间（10 秒），运行本实例程序，显示模块中显示的是译码后的误比特率，同时在工作区中可以看到，变量 DecodedErrorRate 是一个长度为 501 bit 的向量，其中每个元素都表示一帧数据译码之后的误比特率。

为了观测二进制对称信道不同的误比特率对卷积编码器性能的影响，我们需要多次改变 BitErrorRate 的数值，然后重新运行实例程序。下面所示的一段脚本程序（文件名为 project5_2main.m）用于实现上述功能，并且在运行结束之后绘制二进制对称信道误码率与卷积编码信号误码率之间的对数关系图（如图 5-9 所示）。

```
%x 表示二进制对称信道的误比特率的各种取值
x=[0.01 0.02 0.03 0.04 0.05 0.1 0.15 0.2 0.25 0.3 0.4 0.5];
%y 表示卷积编码信号的误码率，它的长度与 x 的长度相等
y=x;
% 对 x 中的每一个元素依次执行仿真
for i=1:length(x)
    % 将二进制对称信道的误比特率设置为 x 的第 i 个元素的数值
    BitErrorRate=x(i);
    % 运行仿真，仿真结果保存在向量 DecodedErrorRate 中
    sim('project5_2');
    % 计算 DecodedErrorRate 的平均值作为卷积编码信号的误码率
    % 为了计算的准确性，舍弃其中的头 100 个元素
    y(i)=mean(DecodedErrorRate(101:501));
end
% 绘制 x 和 y 的对数关系曲线图
loglog(x,y);
```

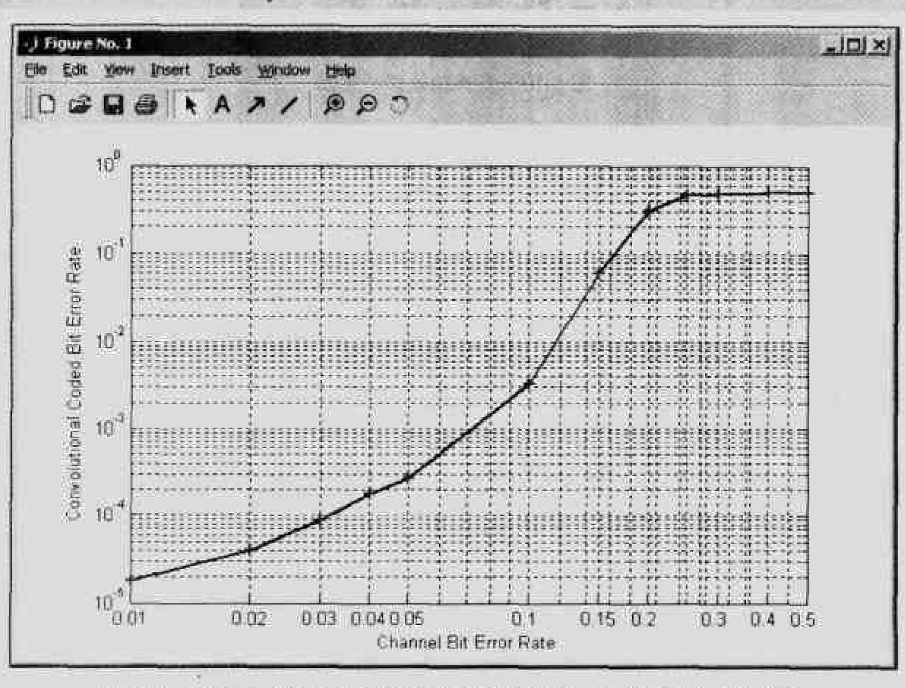



图 5-9 二进制对称信道误码率与卷积编码信号误码率之间的对数关系

从图 5-9 所示中可以看到，当二进制对称信道的误码率小于 20%时，卷积编码信号的误码率都远远低于二进制对称信道误码率；而当二进制对称信道的误码率大于 20%时，卷积编码信号的误码率近似达到了 50%。这说明当信道的误码率高于 20%时，卷积编码基本失去了原有的纠错能力。在实际应用中，移动通信系统的调制和解调能够保证信道的误比特率远远低于 5%。根据我们的仿真结果，当信道的误比特率等于 5%时，卷积编码信号的误码率低于 0.03%，从这里可以看出，卷积编码器具有很强的纠错能力。

## 5.3 多径瑞利衰落信道

瑞利衰落是移动通信系统中的一种相当重要的衰落信道类型，它在很大程度上影响着移动通信系统的质量。在移动通信系统中，发送端和接收端都可能处在不停的运动状态之中，发送端和接收端之间的这种相对运动将产生多普勒频移（Doppler shift）。多普勒频移与运动速度和方向有关，它的计算公式如下：

$$f_d = \frac{v}{\lambda}\cos\theta = \frac{v \times c}{f}\cos\theta \tag{5.4}$$

其中，$v$ 是发送端和接收端的相对运动速度，$\theta$ 是运动方向和发送端与接收端连线之间的夹角（见图 5-10 所示），$\lambda = c/f$ 是载波的波长。

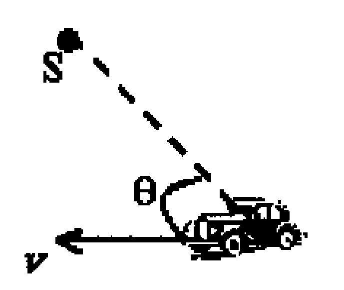



图 5-10　多普勒频移示意图

在多径信道中，发送端发出的信号通过多个发射之后沿多条路经到达接收端，这些路径具有不同的时延和不同的接收强度，它们之间的相互作用就形成了衰落。关于瑞利分布的特点，请参阅 4.1.5 关于瑞利噪声产生器的介绍。

### 5.3.1 多径瑞利衰落信道模块

多径瑞利衰落信道模块（Multipath Rayleigh Fading Channel）实现基带信号的多径瑞利衰落信道仿真，它的输入信号是标量形式或帧格式的复信号。多径瑞利衰落信道模块及其参数设置对话框如图 5-11 所示。

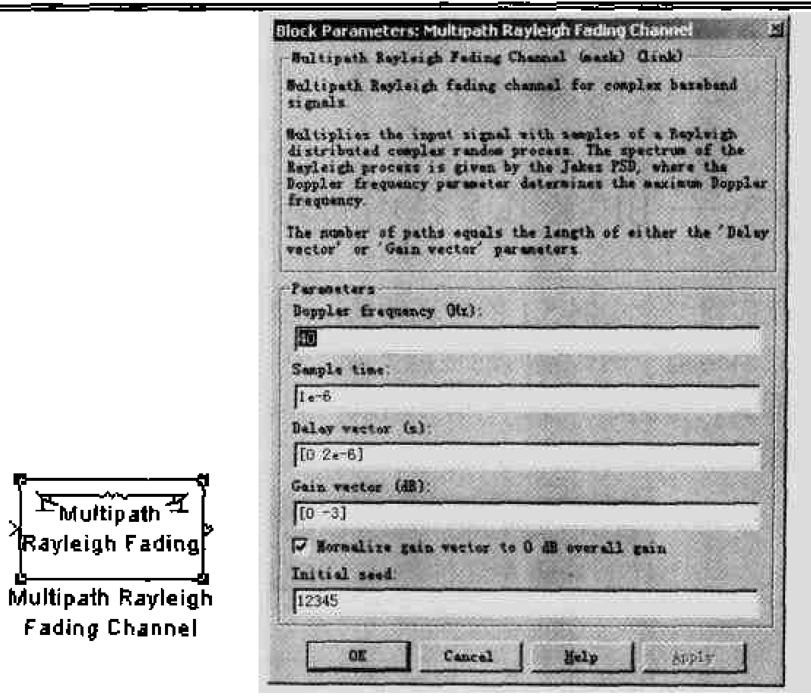



图 5-11　多径瑞利衰落信道模块及其参数设置对话框

在多径瑞利衰落信道模块中，输入信号被延迟一定的时间之后形成多径信号，这些多径信号分别乘于相应的增益，叠加之后就形成了瑞利衰落信号。多径瑞利衰落信道模块的内部结构如图 5-12 所示。

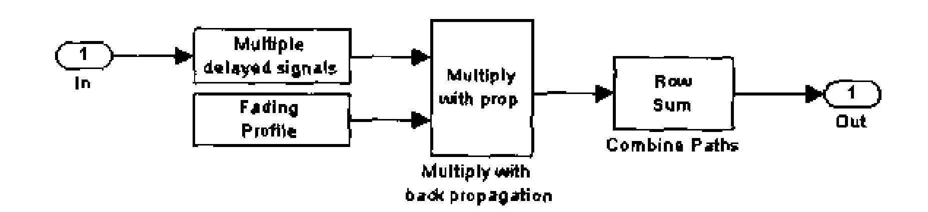



图 5-12　多径瑞利衰落信道模块的内部结构

多径瑞利衰落信道模块主要有以下几个参数。

■　Doppler frequency (Hz)（多普勒频移）

多径瑞利衰落信道模块的最大多普勒频移（单位：Hz）。

■　Sample time（抽样间隔）

多径瑞利衰落信道模块输入信号的抽样间隔（单位：秒）。

■　Delay vector (s)（时延向量）

多径瑞利衰落信道模块各个路径的时延（单位：秒）。

■　Gain vector (dB)（增益向量）

多径瑞利衰落信道模块各个路径的增益（单位：秒）。

■　Normalize gain vector to 0 dB overall gain（增益向量归一化）

当选择本参数前面的复选框之后，多径瑞利衰落信道模块把参数 Gain vector 乘于一个系数作为增益向量，使得所有路径的接收信号强度之和等于 0dB。

■ Initial seed（初始化种子）

多径瑞利衰落信道模块的初始化种子。

## 5.3.2 实例 5.3——BFSK 在多径瑞利衰落信道中的传输性能

在本节中，我们将结合实例 5.1 的仿真结果，比较 BFSK 调制信号在加性高斯白噪声信道和在多径瑞利衰落信道中的传输时的性能差异。图 5-13 是本实例的系统结构框图。从图 5-13 所示中可以看到，实例 5.3 由 3 个部分组成：Source（信源模块）、Channel（信道模块）以及 Sink（信宿模块）。

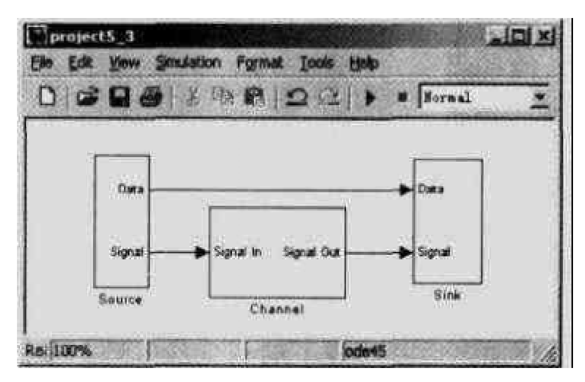



图 5-13 实例 5.3 系统框图

在本实例中，Source（信源模块）产生速率为 10 kbit/s、帧长度为 1 秒的二进制数据 Data，并且通过 BFSK 调制产生调制信号 Signal。调制信号 Signal 通过 Channel（信道模块）后，由 Sink（信宿模块）对其进行解调，并且把解调后的数据与 Source（信源模块）产生的原始数据进行比较，根据比较的结果计算误比特率。最后，Sink（信宿模块）根据信噪比 SNR 与误比特率的对应关系绘制对数曲线图。

■ Source（信源模块）

Source（信源模块）产生数据的速率为 10 kbit/s，每帧的周期为 1 秒，因此，一帧长度等于 10000 bit。图 5-14 所示是 Source（信源模块）的结构框图，它由两部分组成：Random Integer Generator（随机整数产生器）和 M-FSK Modulator Baseband（BFSK 基带调制器）。信源模块与实例 5.1 中的信源模块相同，关于随机整数产生器和 BFSK 基带调制器的参数设置请分别参考表 5-1 和表 5-2。

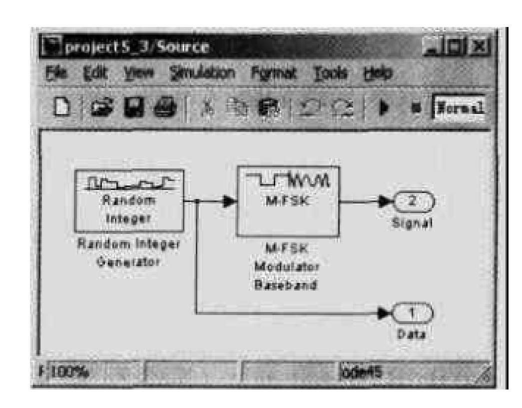



图 5-14 信源模块 Source

■ Sink（信宿模块）

在Sink（信宿模块）中，M-FSK Demodulator Baseband（BFSK基带解调器）对接收信号进行解调，然后通过Error Rate Calculation（误码率计算器）计算该帧的误比特率。误码率计算器产生的数据是一个三维向量，分别表示误码率、误码个数以及信号总数，因此，通过一个Selector（选择器）选择向量的第一个元素作为输出信号。这个输出信号进入To Workspace（工作区）模块，并且保存为变量BitErrorRate。信宿模块的结构框图如图5-4所示。信宿模与实例5.1的信宿模块相同，关于BFSK基带解调器、误码率计算器、选择器和工作区写入模块的参数设置请参照表5-3~表5-6的设置。

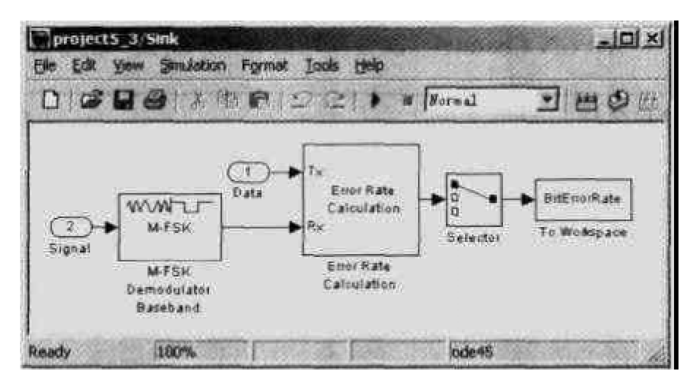



图5-15 信宿模块Sink

■ Channel（信道模块）

在本实例中，Channel（信道模块）由两部分组成，分别是Multipath Rayleigh Fading Channel（多径瑞利衰落信道模块）和AWGN Channel（加性高斯白噪声产生器），如图5-16所示。信道模块首先在BFSK调制信号中引入两径瑞利衰落，然后在衰落信号中叠加高斯白噪声。

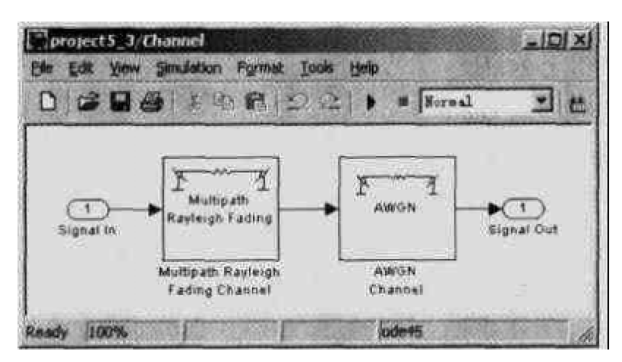



图5-16 信道模块（Channel）

信道模块中的多径瑞利衰落信道模块和加性高斯白噪声产生器的参数设置分别如表5-16和表5-17所示。在这里我们使用了两径瑞利衰落信道，这两径信号的时延分别为0和2μs，它们的相对增益则分别等于0dB和-3dB。

表5-16 Multipath Rayleigh Fading Channel（多径瑞利衰落信道模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Multipath Rayleigh Fading Channel |
| Maximum Doppler shift (Hz) | Fd |

续表

| 参数名称 | 参数值 |
|---|---|
| Sample time | 1/BitRate/SamplesPerSymbol |
| Delay vector (s) | [0 2e-6] |
| Gain vector (dB) | [0 -3] |
| Normalize gain vector to 0 dB overall gain | Checked |
| Initial Seed | 67 |

表5-17　　AWGN Channel（加性高斯白噪声产生器）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | AWGN Channel |
| Initial Seed | 67 |
| Mode | Signal to noise ratio (SNR) |
| SNR (dB) | SNR |
| Input signal power (watts) | 1 |

最后，将运行参数 Simulation | Simulation Parameters | Stop Time 设置为 SimulationTime，并且编写如下的脚本程序（保存为文件 project5_3main.m）：

```
% x 表示信噪比
x=0:15;
% y 表示信号的误比特率，它的长度与 x 相同
y=x;
% BFSK 调制的频率间隔等于 24kHz
FrequencySeparation=24000;
% 信源产生信号的 bit 率等于 10kbit/s
BitRate=10000;
% 仿真时间设置为 10 秒
SimulationTime=10;
% BFSK 调制信号每个符号的抽样数等于 2
SamplesPerSymbol=2;
% 发送端和接收端的相对运动速度（单位是公里/小时）
Velocity=40;
% 光速（单位是米/秒）
LightSpeed=3*10^8;
% 载波频率（单位：Hz）
Frequency=825*10^6;
% 计算载波的波长
WaveLength=LightSpeed/Frequency;
% 根据运动速度和波长计算多普勒频移
```

```
% 注意要把运动速度的单位转换成米/秒
Fd=Velocity*10^3/3600/WaveLength;
% 准备一个空白图
hold off;
% 执行实例 5.1 的仿真程序，得到相应的曲线
project5_1main;
% 保持实例 5.1 的曲线图
hold on;
% 循环执行仿真程序
for i=1:length(x)
    % 信道的信噪比依次取 x 中的元素
     SNR=x(i);
    % 运行仿真程序，得到的误比特率保存在工作区变量 BitErrorRate 中
     sim('project5_3');
    % 计算 BitErrorRate 的均值作为本次仿真的误比特率
     y(i)=mean(BitErrorRate);
end
% 绘制 x 和 y 的关系曲线图，纵坐标采用对数坐标
semilogy(x,y);
```

运行本实例程序，得到一个如图 5-17 所示的曲线图。在这个曲线图中，上面的曲线表示瑞利衰落信道的误比特率，下面的曲线表示实例 5.1 中加性高斯白噪声信道的误比特率，*X* 轴表示的是信噪比 SNR（单位：dB），*Y* 轴表示的是信号的误比特率（对数坐标）。从图中可以看出，当信噪比等于 14 dB 时，加性高斯白噪声信道的误比特率低于$10^{-3}$，而此时多径瑞利衰落信道的误比特率在 5%左右。这样，如果要在瑞利衰落信道中获得与加性高斯白噪声信道相同的传输效果，就需要增加信号的信噪比。在移动通信系统中，瑞利衰落是不可避免的，因此需要采取其他措施来提高通信系统的性能。

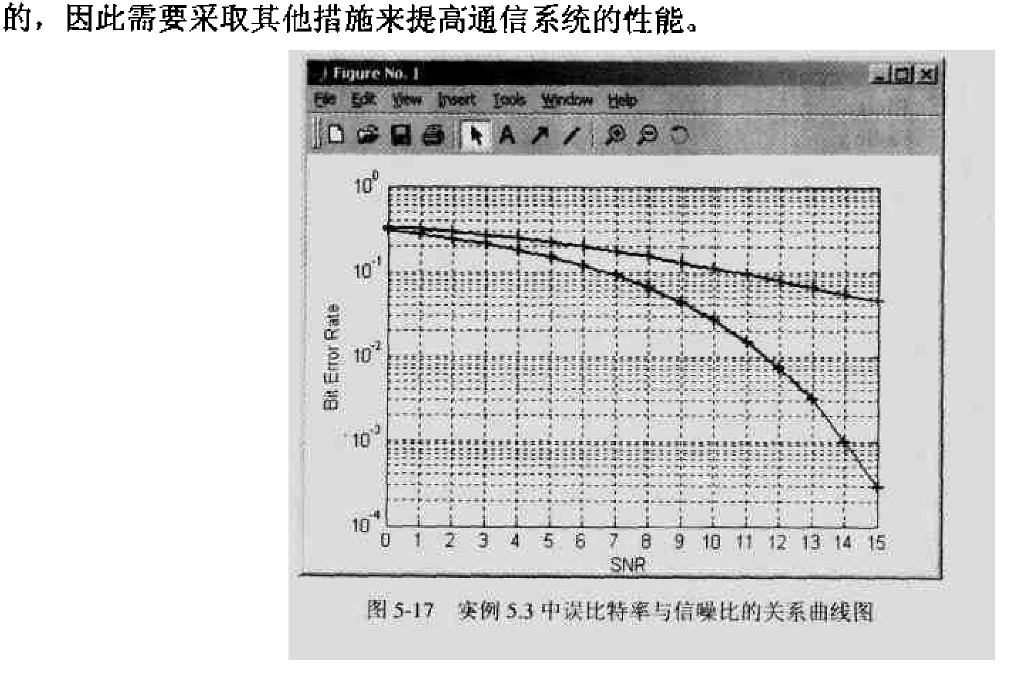



图 5-17　实例 5.3 中误比特率与信噪比的关系曲线图

# 5.4 伦琴衰落信道

在移动通信系统中，如果发送端和接收端之间存在着一条占优势的视距传播路径（LOS，Line-Of-Sight），这种信号就可以模拟成伦琴衰落信道。当发送端和接收端之间既存在着视距传播路径 LOS，又有多条反射路径时，它们之间的信道可以同时用伦琴衰落信道模块和多径瑞利衰落信道来进行仿真。

伦琴衰落信道的一个重要参数是 K 因子，即视距传播路径的能量与反射路径的能量之间的比值。因此，K 因子越大，表示发送端和接收端之间的视距传播路径的能量越强；当 K 因子等于 0 时，发送端和接收端之间不存在视距传播路径，这时候伦琴衰落信道就演变成瑞利衰落信道。关于伦琴分布的特点，请参阅 4.1.5 关于伦琴噪声产生器的介绍。

## 5.4.1 伦琴衰落信道模块

伦琴衰落信道模块（Rician Fading Channel）对基带信号的伦琴衰落信道进行仿真，它的输入信号是标量形式或帧格式的复信号。伦琴衰落信道模块及其参数设置对话框如图 5-18 所示。

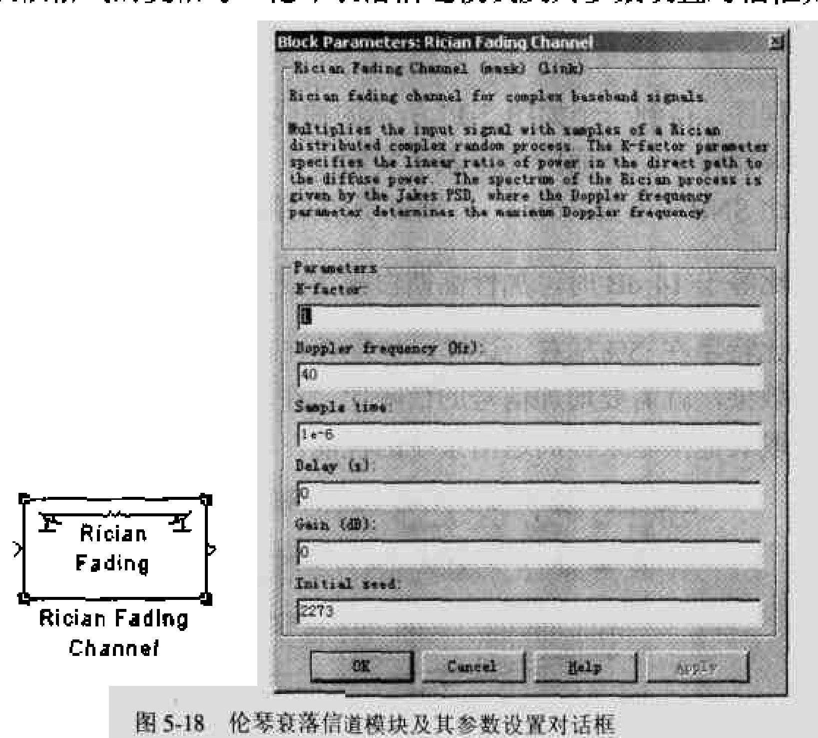



图 5-18 伦琴衰落信道模块及其参数设置对话框

伦琴衰落信道模块主要有以下几个参数：

■ K-factor（K 因子）

伦琴衰落信道模块的 K 因子，它表示视距传播路径（LOS）的能量与其他多径信号的能量之间的比值。

■ Doppler frequency (Hz)（多普勒频移）

伦琴衰落信道模块的多普勒频移（单位：Hz）。

■ Sample time（抽样间隔）

伦琴衰落信道模块输入信号的抽样间隔（单位：秒）。

■ Delay (s)（时延）

伦琴衰落信道模块的时延（单位：秒）。本参数是一个标量，它表示视距传播路径 LOS 的传播时延。

■ Gain (dB)（增益）

伦琴衰落信道模块的增益（单位：dB）。本参数是一个标量，它表示信道对传输信号的总体增益（包括视距传播路径和各种反射路径）。

■ Initial seed（初始化种子）

伦琴衰落信道模块的初始化种子。

伦琴衰落信道模块的内部结构与图 5-12 所示的多径瑞利衰落信道模块相同，这两者的差别在于，在多径瑞利衰落信道模块中，参数 K-factor 被设置为 0。因此，多径瑞利衰落信道模块可以看作是伦琴衰落信道模块的特例。

## 5.4.2 实例 5.4——BFSK 在多径瑞利衰落信道中的传输性能

在前面的讨论中我们知道，当发送端和接收端之间存在着一条视距传播路径 LOS 时，它们之间的信道是伦琴衰落信道。如果同时考虑视距传播和多径衰落，这时候需要用伦琴衰落信道和多径瑞利衰落信道来实现仿真。可以预见的是，由于存在着视距传播路径，信号在伦琴衰落信道中的误比特率性能将优于多径瑞利衰落信道。本节在实例 5.3 的基础上进行扩展，对 BFSK 调制信号在伦琴衰落信道与在多径瑞利衰落信道或加性高斯白噪声信道中的传输性能进行比较。本实例的系统结构框图与实例 5.3 大致相同，如图 5-19 所示。从图 5-19 中可以看到，实例 5.4 由 Source（信源模块）、Channel（信道模块）以及 Sink（信宿模块）3 个部分组成。

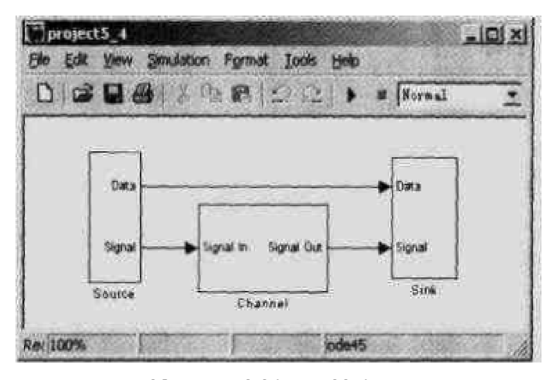



图 5-19 实例 5.4 系统框图

与实例 5.3 相似，本实例的 Source（信源模块）产生速率为 10 kbit/s、帧长度为 1 秒的二进制数据 Data，并且通过 BFSK 调制产生调制信号 Signal。调制信号 Signal 通过 Channel（信道模块）后，在其中引入了伦琴衰落和多径瑞利衰落，并且叠加了高斯白噪声。Sink（信宿模块）对输入信号进行解调，并且把解调后的数据与 Source（信源模块）产生的原始数据进行比较，根据比较的结果计算误比特率。最后，Sink（信宿模块）根据信噪比 SNR 与解调信号的误比特率的对应关系绘制对数曲线图。

本实例中的 Source（信源模块）和 Sink（信宿模块）与实例 5.3 的信源和信宿相同，关于这两个模块的详细情况请参考实例 5.3。与实例 5.3 不同的是，本实例的 Channel（信道模块）由 Rician Fading Channel（伦琴衰落信道模块）、Multipath Rayleigh Fading Channel（多径瑞利衰落信道模块）和 AWGN Channel（加性高斯白噪声产生器）3 部分组成，如图 5-16 所示。信道模块中的多径瑞利衰落信道模块和加性高斯白噪声产生器的参数设置与实例 5.3 中的参数设置相同，如表 5-17 所示。

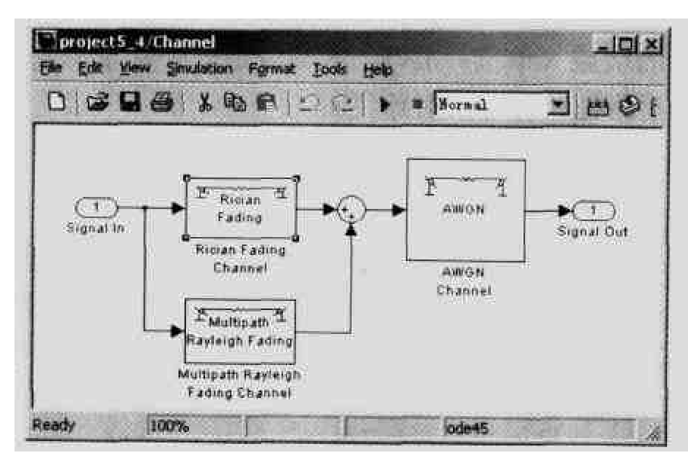



图 5-20　信道模块（Channel）

在信道模块中，BFSK 调制信号分别经过伦琴衰落信道模块和多径瑞利衰落信道模块，这两个模块的输出信号之和就是在存在视距传播路径（LOS）情况下的多径衰落信号。最后，多径衰落信号进入加性高斯白噪声产生器，这时候输出信号就叠加了一定强度的高斯白噪声。本实例中的伦琴衰落信道模块的 K 因子设置为 1，这时候通过视距传播路径（LOS）到达接收端的能量与通过多径衰落到达接收端的能量相同；瑞利衰落信道则模拟了两径衰落的情况，这两径信号的时延分别为 0 和 2μs，它们的相对增益则分别等于 0dB 和-3dB。

**表5-18　　伦琴衰落信道模块（Rician Fading Channel）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Rician Fading Channel |
| K-factor | 1 |
| Maximum Doppler shift (Hz) | Fd |
| Sample time | 1/BitRate/SamplesPerSymbol |
| Delay (s) | 0 |
| Gain (dB) | 0 |
| Initial Seed | 79 |

本实例的运行时间长度由变量 SimulationTime 确定（即将参数 Simulation | Simulation Parameters | Stop Time 设置为 SimulationTime）。跟实例 5.3 相似地，编写 project5_4main.m 脚本程序，其代码如下：

```
% x 表示信噪比
x=0:15;
% y 表示信号的误比特率，它的长度与 x 相同
y=x;
```

```
% BFSK 调制的频率间隔等于 24kHz
FrequencySeparation=24000;
% 信源产生信号的 bit 率等于 10kbit/s
BitRate=10000;
% 仿真时间设置为 10 秒
SimulationTime=10;
% BFSK 调制信号每个符号的抽样数等于 2
SamplesPerSymbol=2;

% 发送端和接收端的相对运动速度（单位是公里/小时）
Velocity=40;
% 光速（单位是米/秒）
LightSpeed=3*10^8;
% 载波频率（单位：Hz）
Frequency=825*10^6;
% 计算载波的波长
WaveLength=LightSpeed/Frequency;
% 根据运动速度和波长计算多普勒频移
% 注意要把运动速度的单位转换成米/秒
Fd=Velocity*10^3/3600/WaveLength;

% 执行实例 5.3 的仿真程序，得到相应的曲线
project5_3main;
% 保持实例 5.3 的曲线图
hold on;

% 循环执行仿真程序
for i=1:length(x)
    % 信道的信噪比依次取 x 中的元素
    SNR=x(i);
    % 运行仿真程序，得到的误比特率保存在工作区变量 BitErrorRate 中
    sim('project5_4');
    % 计算 BitErrorRate 的均值作为本次仿真的误比特率
    y(i)=mean(BitErrorRate);
end

% 绘制 x 和 y 的关系曲线图，纵坐标采用对数坐标
semilogy(x,y);
```

在 MATLAB 工作区中输入命令行“project5_4main”，运行本实例程序，得到一个如图

5-21 所示的曲线图。在图 5-21 中，绿色线条表示伦琴衰落信道中解调信号的误比特率，蓝色线条表示瑞利衰落信道的误比特率，红色线条表示实例 5.1 中加性高斯白噪声信道的误比特率。从图中可以看出，对于相同的信噪比，伦琴衰落信道的误比特性能明显优于多径瑞利衰落信道。一个有趣的现象是，当信号的信噪比低于 6dB 时，伦琴衰落信道的误比特性能甚至好于只存在加性高斯白噪声的信道。

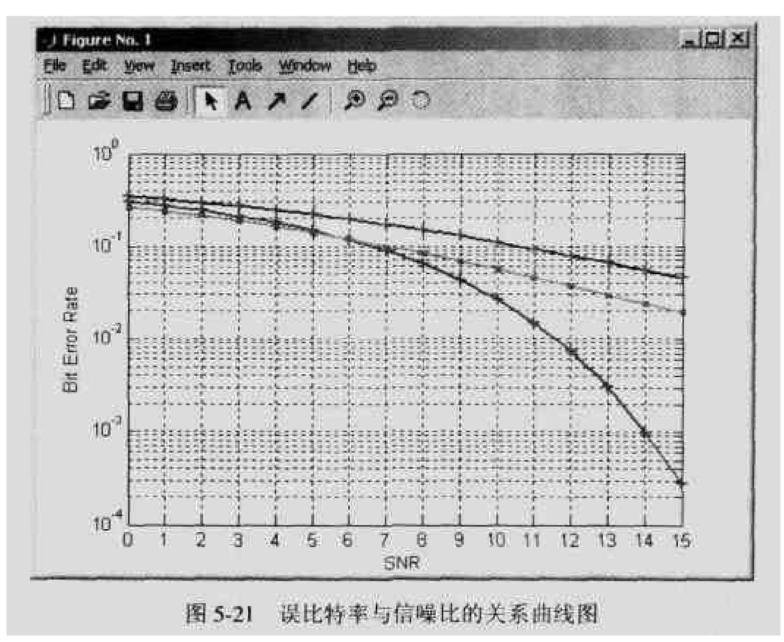



图 5-21　误比特率与信噪比的关系曲线图

在都市环境中，由于众多建筑物的阻挡，移动台与基站之间很难存在一条视距传播路径（LOS），信号主要通过发射和折射到达接收端，仿真过程中一般将信道看作是多径瑞利衰落信道。对于郊区或农村，由于视距传播路径的存在，同样的发射功率可以获得更优的传输效果，这时候采用伦琴衰落信道就显得更恰当。因此，不同的环境需要采用不同的信道模型进行仿真。

## 5.5　射频损耗

射频损耗（RF impairment）是指射频（radio frequency）信号在物理信道或接收机中受到的各种损耗，包括信号在自由空间中传输的损耗，相位和频率偏移，相位噪声，热噪声，以及接收机的非线性作用等等。MATLAB（6.5 以上版本）提供了 6 种模块对这些损耗进行仿真，它们都是在 MATLAB 基本模块的基础上构建的。

### 5.5.1　自由空间路径损耗模块

在无线通信系统中，如果发送端和接收端之间存在着一条没有遮挡的视距传播路径（LOS，Line-Of-Sight），可以使用自由空间路径损耗模型来预测信号的接收功率。假设发射端信号的发射功率为 $P_t$，发射天线的增益等于 $G_t$，载波波长为 $\lambda$，发送端和接收端之间的距

离为 $d$，则在接收端接收到的信号功率可以表示为公式 5.5 所示：

$$P_r(d)=\frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{5.5}$$

其中 $G_r$ 是接收端的接收天线增益，$L$ 是与传输过程无关的各种系统损耗，$L \geq 1$。因此，在自由空间路径损耗模型中，如果不考虑发送端和接收端的天线增益以及各种系统损耗，路径损耗可以表示成信号的发送功率和接收功率之间的比值，如公式 5.6 所示：

$$PL(\mathrm{dB})=10\cdot\log_{10}\frac{P_t}{P_r}=-10\cdot\log_{10}\left(\frac{\lambda^2}{(4\pi)^2 d^2}\right) \tag{5.6}$$

自由空间路径损耗模块（Free Space Path Loss）对信号在自由空间中传输的特点进行模拟，它把输入信号的幅度降低一定的增益，这个增益可以由用户指定，也可以通过发送端和接收端之间的距离以及载波的频率计算出来。自由空间路径损耗模块及其参数设置对话框如图 5-22 所示。

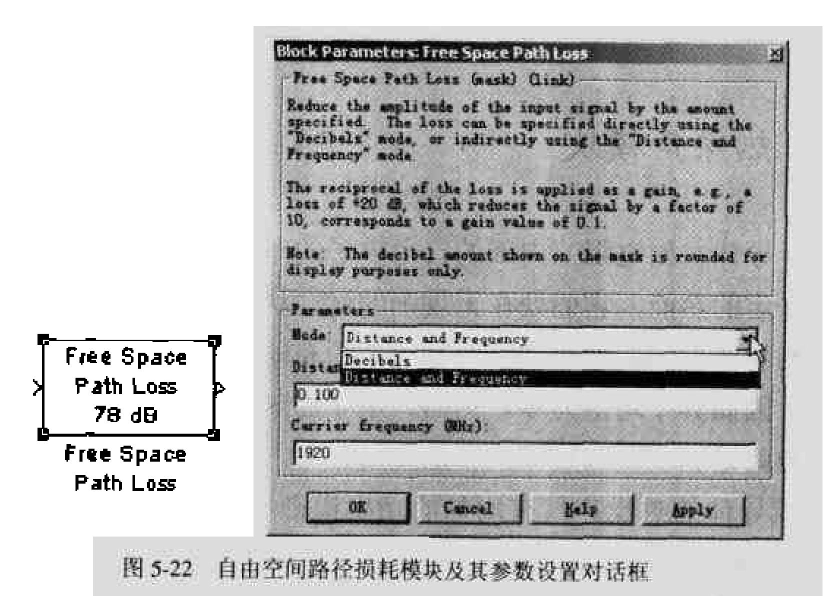



图 5-22　自由空间路径损耗模块及其参数设置对话框

图 5-23 所示是自由空间路径损耗模块的内部结构。从图中可以看到，自由空间路径损耗模块首先根据发送端和接收端之间的距离以及载波的频率计算得到相应的传输损耗 $PL(\mathrm{dB})$（或者是由用户来指定这个传输损耗），然后计算对应的幅度增益 $PL(\mathrm{dB})/2$，最后通过一个增益器（Gain）把信号的幅度降低 $PL(\mathrm{dB})/2$。

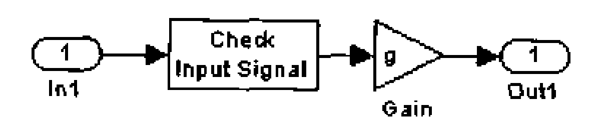



图 5-23　自由空间路径损耗模块的内部结构

自由空间路径损耗模块主要有以下几个参数。

■ Mode（模式）

自由空间路径损耗模块的工作模式。当本参数设置为 Decibels，用户可以指定信号损耗的增益强度；当本参数设置为 Distance and Frequency 时，用户可以指定发送端和接收端之间的距离以及载波的频率，由 MATLAB 根据自由空间传输的特点计算相应的增益。

■ Loss (dB)（路径损耗）

自由空间路径损耗模块的路径损耗（单位：dB）。本参数只在 Mode 设置为 Decibels 时有效。

■ Distance (km)（距离）

发送端和接收端之间的距离（单位：公里）。本参数只在 Mode 设置为 Distance and Frequency 时有效。

■ Carrier frequency (MHz)（载波频率）

输入信号的载波频率（单位：MHz）。本参数只在 Mode 设置为 Distance and Frequency 时有效。

## 5.5.2 接收机热噪声模块

热噪声是由于导体（如电阻）内部的自由电子的布朗运动引起的噪声。导体中的每一个自由电子都具有一定的热能，它们处在无规则的运动状态中，并且和其他粒子不停地发生碰撞，呈现出随机的和曲折的布朗运动。所有电子的布朗运动形成通过导体的电流，这种电流的方向是随机的，因而其平均值为零。然而，电子的这种随机运动还会产生一个交流电流成分，这个交流成分就是热噪声。热噪声服从高斯分布，并且在从直流到微波（$10^{13}$Hz）的频率范围内，电阻或导体的热噪声具有均匀的功率谱密度$2kTG$，其中$k$为玻尔兹曼常数，且$k=1.3805\times10^{-23}(J/K)$，$T$为热噪声源的绝对温度，$G$为导体的电导。

接收机热噪声模块（Receiver Thermal Noise）的输入信号是复数形式的基带信号，它在输入信号中加入热噪声，模拟接收机的热噪声效果。接收机热噪声模块及其参数设置对话框如图 5-24 所示。

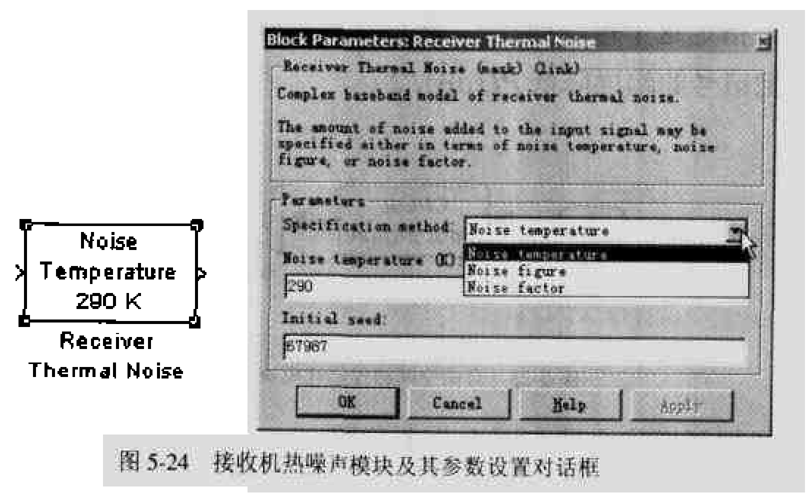



图 5-24 接收机热噪声模块及其参数设置对话框

接收机热噪声模块主要有以下几个参数。

■ Specification method（表示方式）

接收机热噪声模块的表示方式：Noise temperature、Noise figure 或 Noise factor。

■ Noise temperature (K)（热噪声温度）

接收机热噪声模块的热噪声温度（单位：开尔文）。本参数在 Specification method 设置为 Noise temperature 时有效。

■ Noise figure (dB)（噪声指数）

接收机热噪声模块的噪声指数（单位：dB）。本参数在 Specification method 设置为 Noise figure 时有效。噪声指数是相对于开氏温度 290 度时的噪声强度（dB 值）。噪声指数等于 0dB 时表示无噪声系统。

■ Noise factor（噪声系数）

接收机热噪声模块的噪声系数。本参数在 Specification method 设置为 Noise factor 时有效。

■ Initial seed（初始化种子）

接收机热噪声模块的初始化种子。

图 5-25 所示是接收机热噪声模块的内部结构。接收机热噪声模块根据设定的关于噪声强度的参数计算得到一个数值，这个数值就是加性高斯白噪声模块（AWGN Noise Temp）的方差。AWGN Noise Temp 产生一个加性高斯白噪声作为接收机的热噪声，叠加到输入信号后形成接收机热噪声模块的输出信号。

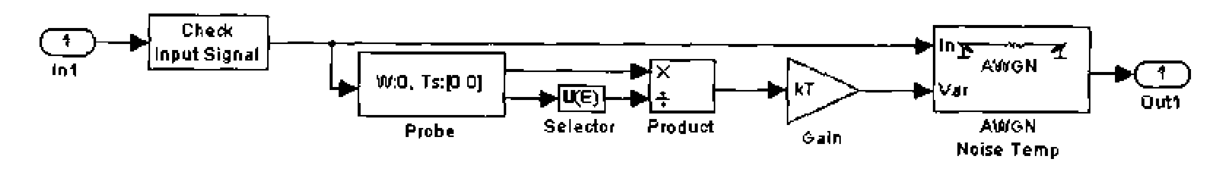



图 5-25 接收机热噪声模块的内部结构

## 5.5.3 相位噪声模块

相位噪声模块（Phase Noise）的输入信号是复数形式的基带信号，它通过 AWGN 模块产生一个加性高斯白噪声信号，并且把这个加性高斯白噪声信号当作相位噪声叠加到输入信号中。相位噪声模块及其参数设置对话框如图 5-26 所示。

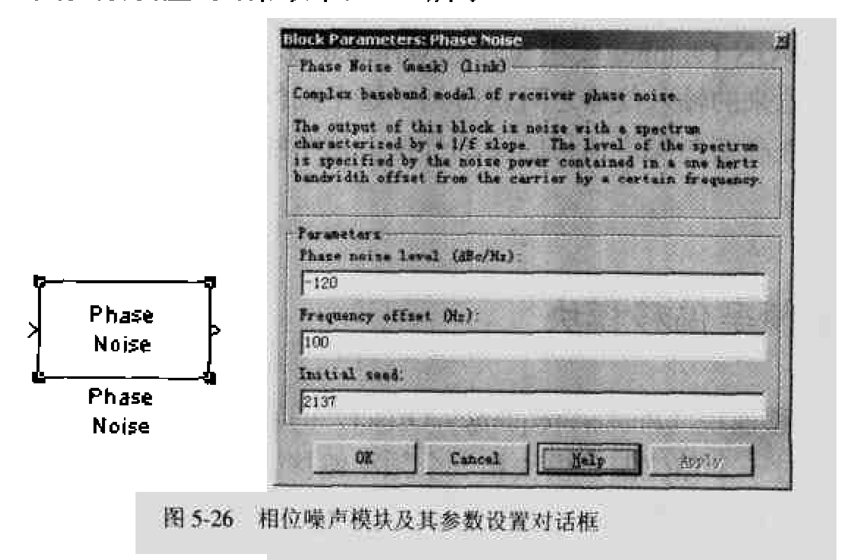



图 5-26 相位噪声模块及其参数设置对话框

相位噪声模块主要有以下 3 个参数。

■ Phase noise level (dBc/Hz)（相位噪声强度）

相位噪声模块产生的相位噪声的强度（单位：dBc/Hz）。

■ Frequency offset (Hz)（频率偏移）

相位噪声模块的频率偏移（单位：Hz）。

■ Initial seed（初始化种子）

相位噪声模块的初始化种子。本参数是一个正整数。

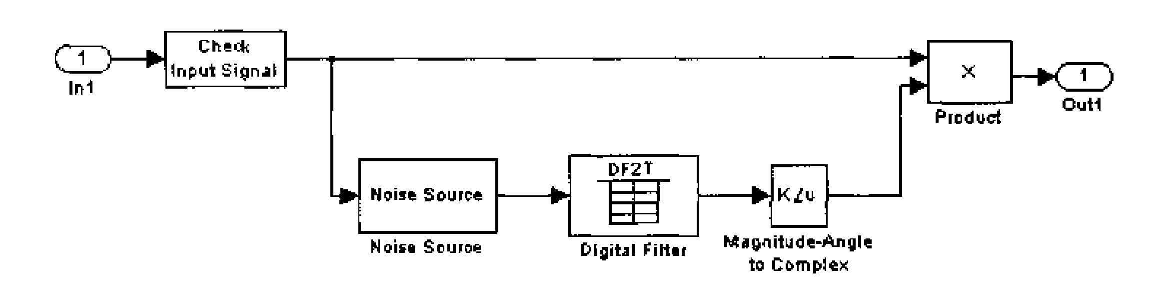



图 5-27　相位噪声模块的内部结构

在相位噪声模块中，它通过 Noise Source 模块产生一个加性高斯白噪声，这个白噪声信号经过数字滤波之后成为噪声的角度分量叠加到输入信号中，如图 5-27 所示。其中 Magnitude-Angle to Complex 模块把经过数字滤波之后的信号转换成复数的角度分量（即参数 Input 设置为 Angle，并且参数 Magnitude 等于 1)。

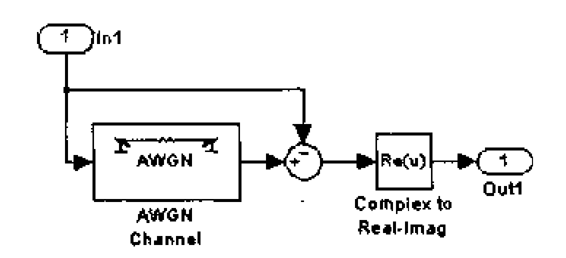



图 5-28　相位噪声模块中的 Noise Source 模块的内部结构

图 5-28 所示是相位噪声模块中的 Noise Source 模块的内部结构。在 Noise Source 模块中，输入信号经过 AWGN Channel 模块（方差 Variance 设置为 2）后叠加了一个加性高斯白噪声信号，从中减去原来的输入信号之后就得到了这个加性高斯白噪声信号。Noise Source 模块的输出信号是加性高斯白噪声信号的实部分量。这个实部分量经过数字滤波之后就成为相位噪声。

## 5.5.4　相位/频率偏移模块

相位/频率偏移模块（Phase/Frequency Offset）的输入信号是复数形式的基带信号，这个基带信号分别经过一个相位偏移模块和一个频率偏移模块的处理之后，产生带有特定的相位和频率偏移的基带信号。相位/频率偏移模块及其参数设置对话框如图 5-29 所示。

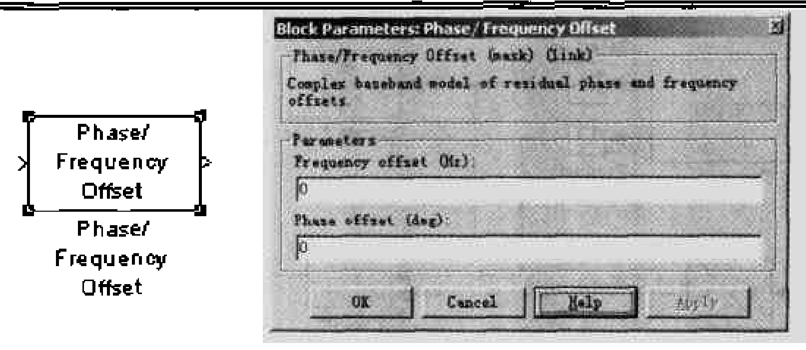



图 5-29　相位/频率偏移模块及其参数设置对话框

相位/频率偏移模块有以下两个参数：

■　Frequency offset (hz)（频率偏移）

相位/频率偏移模块的频率偏移（单位：Hz）。

■　Phase offset (deg)（相位偏移）

相位/频率偏移模块的角度偏移（单位：度）。

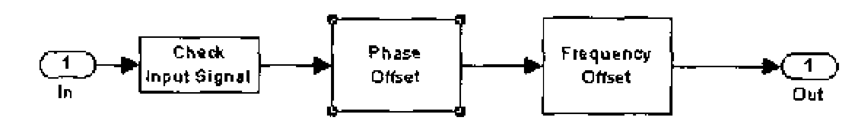



图 5-30　相位/频率偏移模块的内部结构

在相位/频率偏移模块中，输入信号在通过一致性检查之后，由 Phase Offset 模块产生相位偏移，然后由 Frequency Offset 模块产生频率偏移，如图 5-30 所示。

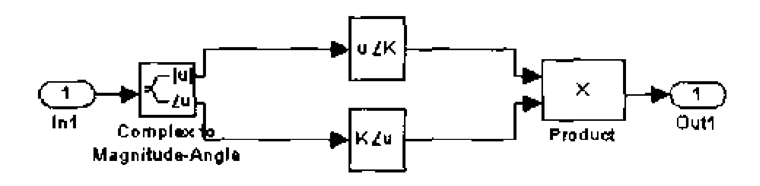



图 5-31　相位/频率偏移模块中的 Phase Offset 模块的内部结构

图 5-31 所示是相位/频率偏移模块中的 Phase Offset 模块的内部结构图。在 Phase Offset 模块中，输入信号 $x=Ae^{j\theta}$ 被分成幅度 $A$ 和相位 $\theta$ 两个分量分别进行处理，输出信号 $y$ 就是这两个分量的乘积。假设相位偏移为　（注意这里使用的单位是角度而不是弧度），则幅度分量 $A$ 经过处理之后变成 $Ae^{j\ \times\pi/180}$，相位分量 $\theta$ 保持不变，仍然是 $e^{j\theta}$，这时候产生的输出信号就是 $y=Ae^{j(\theta+\ \times\pi/180)}$。

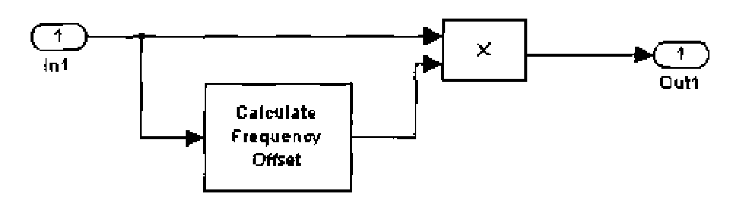



图 5-32　相位/频率偏移模块中的 Frequency Offset 模块的内部结构

Frequency Offset 模块把加入了相位偏移的输入信号通过 Calculate Frequency Offset 模块产生一个频率偏移信号，然后把这个信号与模块的输入信号进行相乘之后，就得到相位/频率偏移模块的输出信号，如图 5-32 所示。Calculate Frequency Offset 模块如图 5-33 所示。

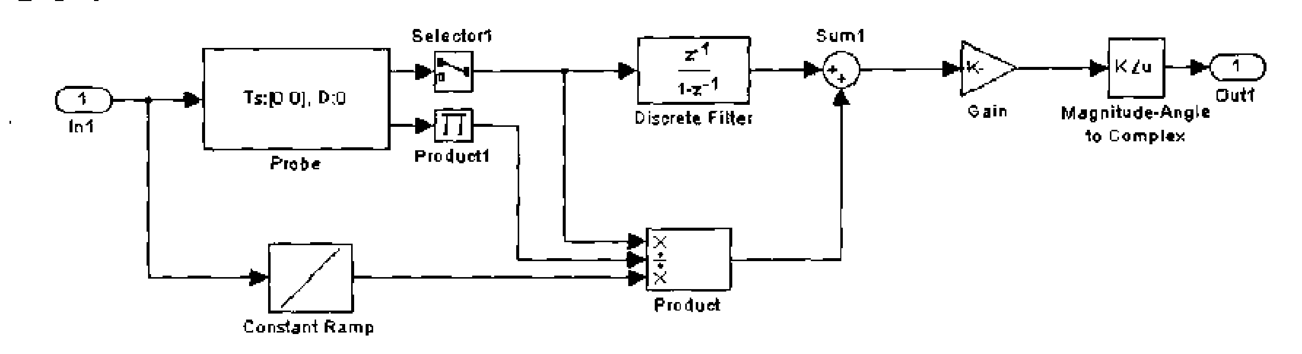



图 5-33 Calculate Frequency Offset 模块的内部结构

假设相位/频率偏移模块的频率偏移参数设置为α，则 Calculate Frequency Offset 模块的输入信号在经过一定的变换之后乘于一个增益 $2\pi\alpha$，然后这个信号被当做角度分量构成幅度为 1 的复数信号。这个复数信号输出到 Frequency Offset 模块中，与 Frequency Offset 模块的输入信号相乘，就得到了带有频率偏移的输出信号。

## 5.5.5 I/Q 支路失衡模块

I/Q 支路失衡一般是由无线传输信道对 I 支路信号和 Q 支路信号的不同损耗引起的。I/Q 支路失衡模块（I/Q Imbalance）是对信号的 I/Q 支路失衡的模拟，它的输入信号是复数形式的基带信号，这个信号的 I 支路分量和 Q 支路分量在不同的幅度和相位以及直流分量的影响之下改变了原先的数值。I/Q 支路失衡模块及其参数设置对话框如图 5-34 所示。

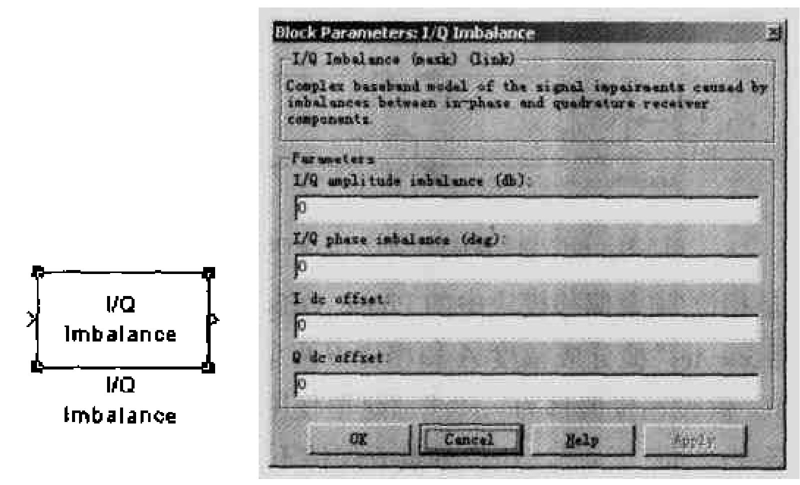



图 5-34 I/Q 支路失衡模块及其参数设置对话框

I/Q 支路失衡模块主要有以下几个参数。

■ I/Q amplitude imbalance (dB)（幅度失衡）

I/Q 支路失衡模块的幅度失衡参数 $X$（单位：dB）。当 $X=0$ 时，I 支路信号和 Q 支路信号的幅度保持不变；当 $X>0$ 时，I 支路信号的幅度增加 $X/2$ dB，Q 支路信号的幅度则降低 $X/2$dB；当 $X<0$ 时，I 支路信号的幅度降低 $X/2$ dB，Q 支路信号的幅度增加 $X/2$ dB。

■ I/Q phase imbalance (deg)（相位失衡）

I/Q 支路失衡模块的相位失衡参数$\theta$（单位：度）。当$\theta=0$时，I 支路信号和 Q 支路信号的相位保持不变；当$\theta\neq0$时，I 支路信号的相位等于$-\theta\cdot\pi/360$（弧度），Q 支路信号的相位

等于$\pi/2+\theta\cdot\pi/360$（弧度）。

■　I dc offset（I 支路直流偏移）

I/Q 支路失衡模块中 I 支路信号的直流偏移。

■　Q dc offset（Q 支路直流偏移）

I/Q 支路失衡模块中 Q 支路信号的直流偏移。

图 5-35 所示是 I/Q 支路失衡模块的内部结构图。I/Q 支路失衡模块输入的复信号首先被分成实部和虚部，它们分别对应于输入信号的 I 支路分量和 Q 支路分量。这两个分量分别通过增益器 Igain 和 Qgain 进行幅度变换，然后通过两个 Magnitude-Angle to Complex 模块（即图 5-35 所示中标有 Phase Imbalance 的模块）转换成复数信号，这两个分量之和与 I 支路直流分量和 Q 支路直流分量相加之后就得到 I/Q 支路失衡模块的输出信号。

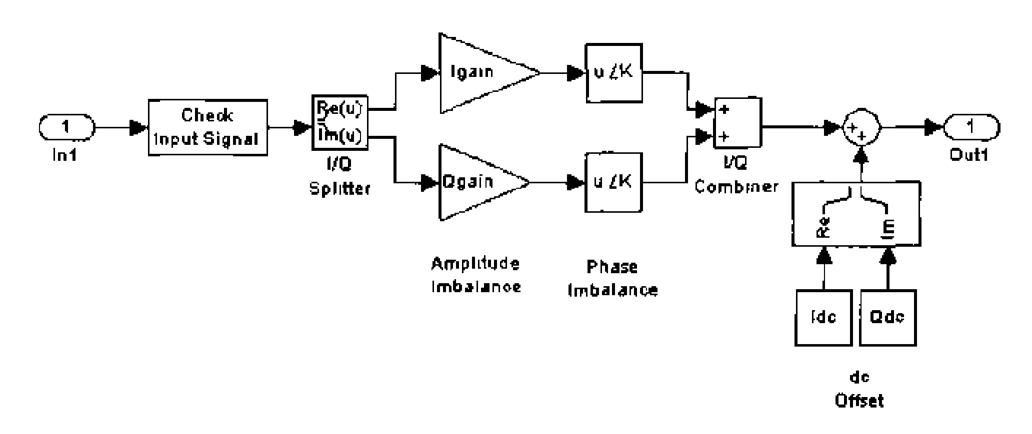



图 5-35　I/Q 支路失衡模块的内部结构

## 5.5.6　无记忆非线性模块

无记忆非线性模块（Memoryless Nonlinearity）对输入的基带复信号实施无记忆非线性处理，这种处理过程一般用于对无线通信系统中接收机的射频损耗进行仿真。在 MATLAB 中，无记忆非线性模块有 5 种处理方式，即 Cubic polynomial、Hyperbolic tangent、Saleh model、Ghorbani model 以及 Rapp model。无记忆非线性模块及其参数设置对话框如图 5-36 所示。

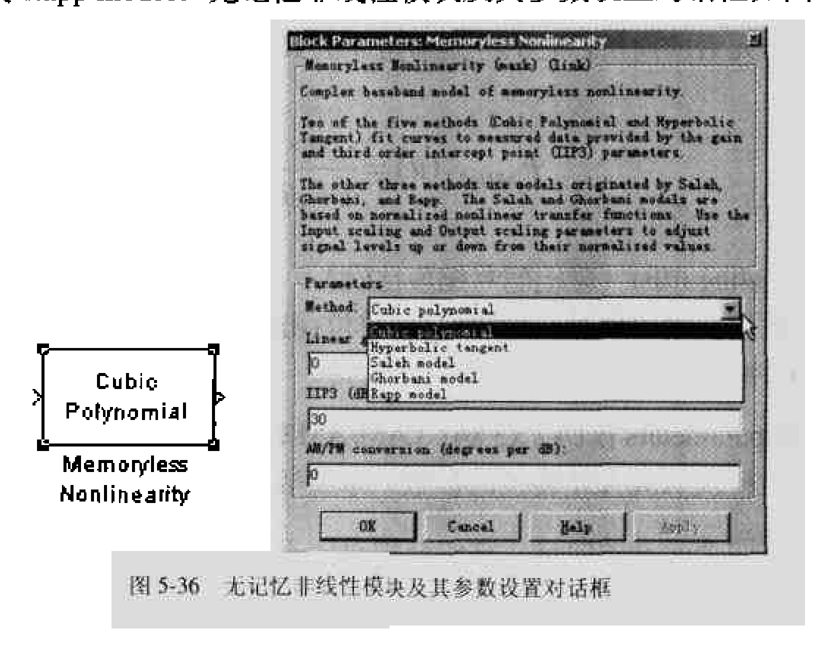



图 5-36　无记忆非线性模块及其参数设置对话框

无记忆非线性模块主要有以下几个参数。

■ Method（模式）

无记忆非线性模块的工作模式：Cubic polynomial、Hyperbolic tangent、Saleh model、Ghorbani model 或 Rapp model。

■ Linear gain (db)（线性增益）

无记忆非线性模块的线性增益（单位：dB）。本参数在 Method 设置为 Cubic polynomial、Hyperbolic tangent 或 Rapp model 时有效。

■ IIP3 (dBm)（三阶截距）

无记忆非线性模块的三阶截距(单位:dB毫瓦)。本参数在 Method 设置为 Cubic polynomial 或 Hyperbolic tangent 时有效。

■ AM/PM conversion (degrees per dB)（AM/PM 转换参数）

无记忆非线性模块 AM/PM 转换的参数（单位：度/dB）。本参数在 Method 设置为 Cubic polynomial 或 Hyperbolic tangent 时有效。

■ Input scaling (dB)（输入信号缩放比例）

无记忆非线性模块输入信号的缩放比例(单位:dB)。本参数在 Method 设置为 Saleh model 或 Ghorbani model 时有效。

■ AM/AM parameters [alpha beta]（AM/AM 转换参数）

无记忆非线性模块 AM/AM 转换的参数。本参数在 Method 设置为 Saleh model 时有效。它是一个具有两个元素的向量，它们分别对应于公式 5.7 中的 $\alpha$ 和 $\beta$。

$$F_{\mathrm{AM/AM}}(u)=\frac{\alpha\times u}{1+\beta\times u^2} \tag{5.7}$$

■ AM/PM parameters [alpha beta]（AM/PM 转换参数）

无记忆非线性模块 AM/PM 转换的参数。本参数在 Method 设置为 Saleh model 时有效。它是一个具有两个元素的向量，它们分别对应于公式 5.8 中的 $\alpha$ 和 $\beta$。

$$F_{\mathrm{AM/AM}}(u)=\frac{\alpha\times u^2}{1+\beta\times u^2} \tag{5.8}$$

■ Output scaling (dB)（输出信号缩放比例）

无记忆非线性模块输出信号的缩放比例(单位:dB)。本参数在 Method 设置为 Saleh model 或 Ghorbani model 时有效。

■ AM/AM parameters [x1 x2 x3 x4]（AM/AM 转换参数）

无记忆非线性模块 AM/AM 转换的参数。本参数在 Method 设置为 Ghorbani model 时有效。它是一个具有四个元素的向量，它们分别对应于公式 5.9 中的 $x_1$、$x_2$、$x_3$ 和 $x_4$。

$$F_{\mathrm{AM/AM}}(u)=\frac{x_1u^{x_2}}{1+x_3u^{x_2}}+x_4u \tag{5.9}$$

■　AM/PM parameters [y1 y2 y3 y4]（AM/PM 转换参数）

无记忆非线性模块 AM/PM 转换的参数。本参数在 Method 设置为 Ghorbani model 时有效。它是一个具有 4 个元素的向量，它们分别对应于公式 5.10 中的 $y_1$、$y_2$、$y_3$ 和 $y_4$。

$$F_{\mathrm{AM/PM}}(u)=\frac{y_1u^{y_2}}{1+y_3u^{y_2}}+y_4u \tag{5.10}$$

■　Smoothness factor（平滑因子）

无记忆非线性模块的平滑因子。本参数在 Method 设置为 Rapp model 时有效。在 Rapp model 模型中，AM/AM 转换函数如公式 5.11 所示，公式中的参数 S 就是平滑因子。

$$F_{\mathrm{AM/AM}}(u)=\frac{u}{\left(1+\left(\frac{u}{O_{sat}}\right)^{2S}\right)^{1/2S}} \tag{5.11}$$

■　Output saturation level（输出信号上限）

无记忆非线性模块输出信号的上限。本参数在 Method 设置为 Rapp model 时有效，它对应于公式 5.11 中的参数 $O_{\mathrm{sat}}$。

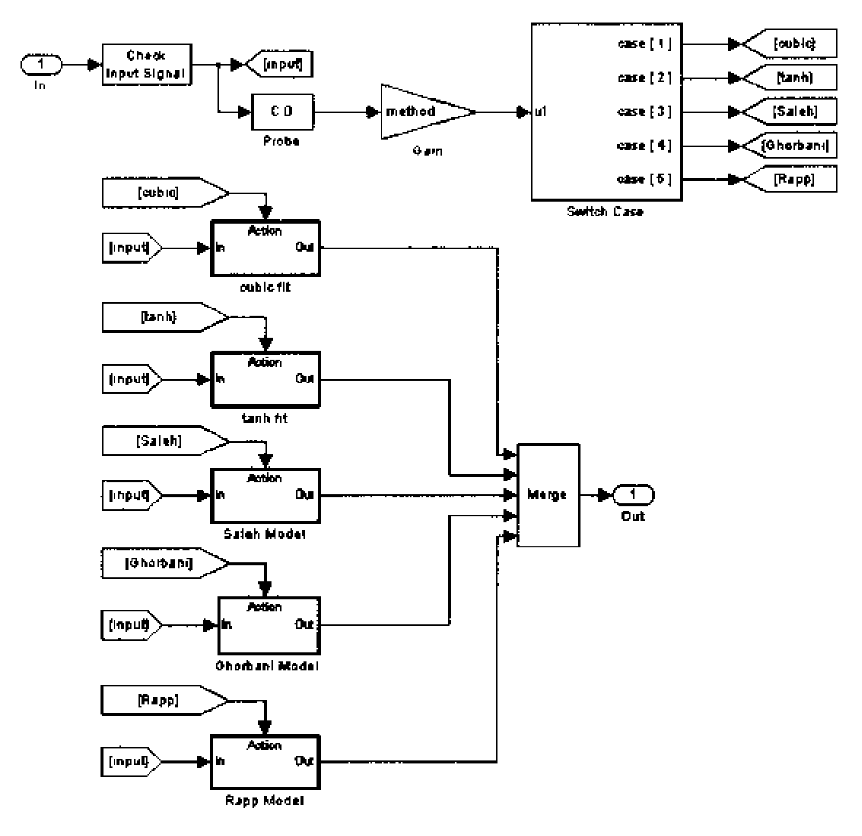



图 5-37　无记忆非线性模块的内部结构

图 5-37 所示是无记忆非线性模块的内部结构。在无记忆非线性模块中，输入信号根据模块中设置的模式 Method 进入相应的处理模块（即 cubic fit、tanh fit、Saleh Model、Ghorbani model 或 Rapp Model 模块）。这 5 种模块对输入信号有相似的处理流程，其中 cubic fit 模块的内部结构如图 5-38 所示。

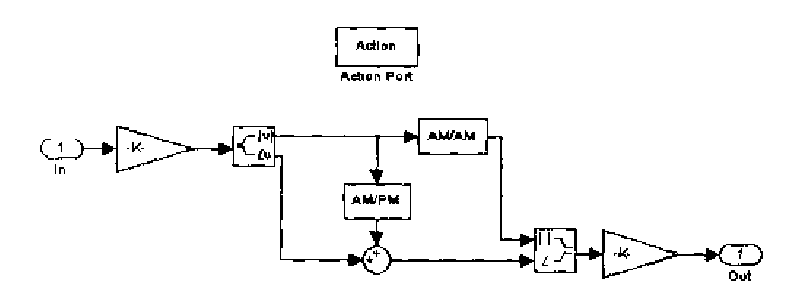



图 5-38　无记忆非线性模块中的 cubic fit 模块的内部结构

在 cubic fit 模块中，输入信号乘于一个增益之后被分成幅度和相位两部分。这两个部分通过不同的处理过程之后重新组合在一起，并且在乘于另外一个增益之后形成输出信号。幅度分量由 AM/AM 模块进行处理，同时幅度分量还要经过 AM/PM 模块产生一个信号，这个信号与相位分量相加之后就得到处理之后的相位分量。不同的处理模式有不同的 AM/AM 模块和 AM/PM 模块，它们产生不同的输出信号。

# 第6章　信源编码

通信网络的最初目的是用来传送话音的，它把人类发出的语音信号转换成电磁信号，通过线缆或电磁波传送到远方，远方的接收端再通过与发送端相反的变换把接收到的电磁信号还原为语音信号，这就是语音通信的基本原理。

对语音信号来说，既可以采用模拟调制方式，也可以采用数字调制方式。模拟调制直接对语音信号进行幅度调制或频率调制，它的优点是实现简单，但是它也存在着诸多的问题，逐渐地已经被数字调制方式所取代。在数字调制过程中，语音信号首先通过一个量化编码器，把连续的语音信号转换成离散的数字信号，然后再对这些数字信号进行调制。接收端接收到信号后，通过相反的变换过程得到数字序列，再通过量化解码器把数字序列还原成语音信号。本章将讨论如何把模拟信号与数字信号的相互转化，即信源编码和解码的原理，以及几种常用的量化编码器和解码器。

## 6.1　压缩和扩展

模拟信号的量化由两种方式：均匀量化和非均匀量化。均匀量化把输入信号的取值范围等距离地分割成若干个量化区间，无论抽样值大小如何，量化噪声的均方根都固定不变，因此实际过程中大多采用非均匀量化。比较常用的两种非均匀量化的方法是 A 律压缩和 μ 律压缩。美国采用 μ 律压缩和扩展，我国和欧洲各国均采用 A 律压缩和扩展。本节我们介绍 A 律压缩及其逆过程——A 律扩展，μ 律压缩和扩展的原理将在下一节里介绍。

A 率压缩是一种非均匀量化方法。假设输入信号为 $x$，输出信号为 $y$，则 A 律压缩满足公式 6.1。

$$y=\begin{cases}\dfrac{A|x|}{1+\log A}\mathrm{sgn}(x), 0\le |x| \le \dfrac{V}{A} \\ \dfrac{V(1+\log(A|x|/V))}{1+\log A}\mathrm{sgn}(x), \dfrac{V}{A}<|x|\le V\end{cases} \tag{6.1}$$

其中，$A$ 是 A 律压缩的参数，$V$ 是输入信号 $x$ 的最大值。

### 6.1.1　A 律压缩模块

A 律压缩模块（A-Law Compressor）对输入信号实施 A 率压缩，产生非均匀量化信号。图 6-1 所示是 A 律压缩模块及其参数设置对话框。

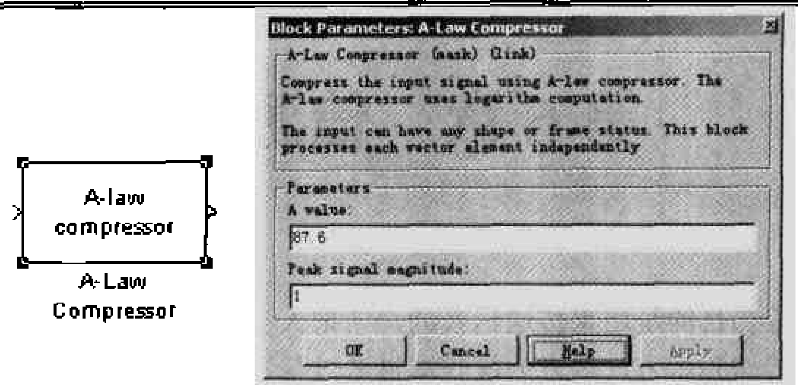



图 6-1　A 律压缩模块及其参数设置对话框

A 律压缩模块主要有以下两个参数。

■　A value（参数 A）

A 律压缩的参数，对应于公式 6.1 中的参数 $A$。

■　Peak signal magnitude（输入信号的峰值）

输入信号的最大值，对应于公式 6.1 中的参数 $V$。

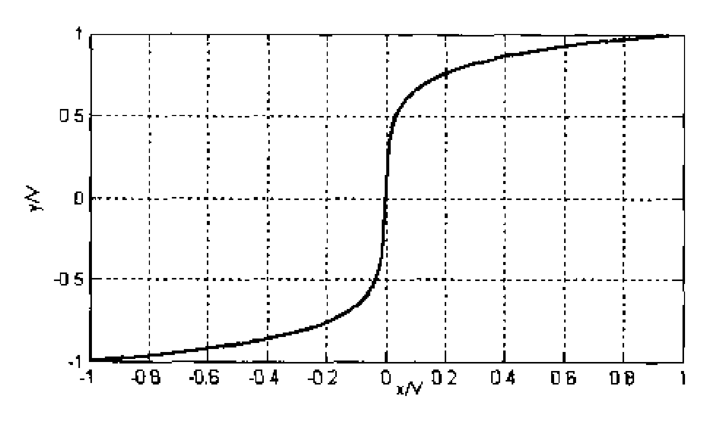



图 6-2　A=87.6 时 A 律压缩的曲线

图 6-2 所示是当 A 等于 87.6 时根据公式 6.1 计算得到的 A 律压缩曲线，图中的横坐标和纵坐标都是归一化后的数值。从图中可以看到，输出信号的幅度范围与输入信号相同。当输入信号的幅度较小时，输出信号的变化幅度较大；而当输入信号的幅度较大时，输出信号的变化则相对变小，这就是非均匀量化的特征，它能有效的降低量化噪声。

## 6.1.2　A 律扩展模块

A 律扩展模块（A-Law Expander）实施与 A 律压缩模块相反的过程，公式 6.2 所示是 A 律扩展的工作原理。

$$x=\begin{cases}\dfrac{y(1+\ln A)}{A}, 0\le|y|\le\dfrac{V}{1+\ln A}\\ e^{|y|(1+\ln A)/V-1}\dfrac{V}{A}\text{sgn}(y), \dfrac{V}{1+\ln A}<|y|\le V\end{cases}\tag{6.2}$$

公式 6.2 所示中的函数实际上是公式 6.1 的逆函数。A 律扩展模块及其参数设置对话框如

图 6-1 所示。

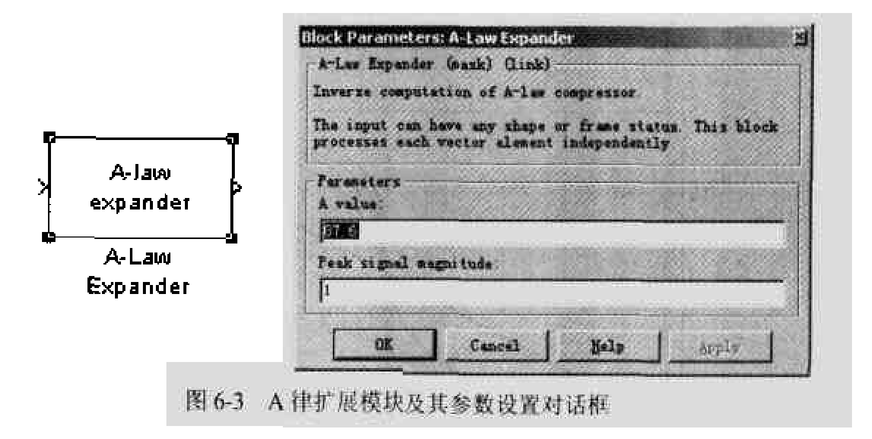



图 6-3　A 律扩展模块及其参数设置对话框

A 律扩展模块主要有以下两个参数。

■　A value（参数 $A$）

A 律扩展的参数，对应于公式 6.2 中的参数 $A$。

■　Peak signal magnitude（输入信号的峰值）

输入信号的最大值，对应于公式 6.2 中的参数 $V$。

## 6.1.3　μ 律压缩模块

μ 律压缩和扩展是我国和欧洲各国广泛采用的非均匀量化算法。对于 μ 律压缩的输入信号 $x$，输出信号 $y$ 由下列公式确定：

$$y = \frac{V\ln(1+\mu|x|/V)}{\ln(1+\mu)}\mathrm{sgn}(x) \tag{6.3}$$

其中参数 $\mu$ 是 μ 律压缩参数，$V$ 是输入信号的最大值，$\ln(x)$表示取 $x$ 的自然对数。

μ 律压缩模块（Mu-Law Compressor）对输入信号实施 μ 律压缩，产生非均匀量化信号。μ 律压缩模块及其参数设置如图 6-4 所示。

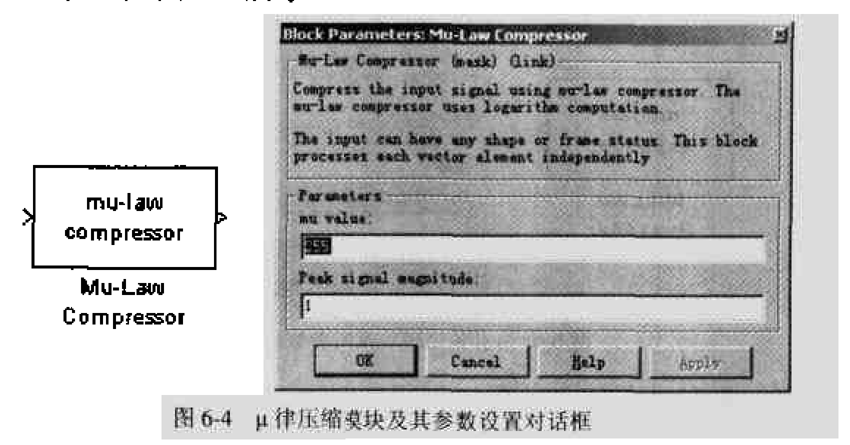



图 6-4　μ 律压缩模块及其参数设置对话框

μ 律压缩模块主要有以下两个参数。

■　mu value（参数 $\mu$）

$\mu$ 律压缩参数，对应于公式 6.3 中的参数 $\mu$。

■ Peak signal magnitude（输入信号的峰值）

输入信号的最大值，对应于公式 6.3 中的参数 $V$。

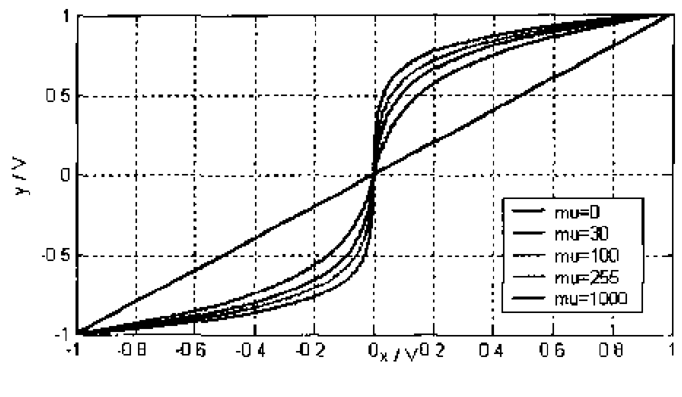



图 6-5 μ 律压缩曲线

图 6-5 所示是当 $\mu$ 分别等于 0、30、100、255 以及 1000 时 μ 律压缩的曲线。从图中可以看到，当 $\mu$ 等于 0 时，压缩特性是通过原点的一条直线，没有压缩效果；当 $\mu$ 值增大时，压缩效果明显加强，这时候能够有效地改善小信号的性能。一般情况下，当 $\mu$ 大于 100 时，压缩器的效果就比较理想了。通常 $\mu$ 的取值是 255。

## 6.1.4 μ 律扩展模块

μ 律扩展是 μ 律压缩的逆过程，当输入信号为 $y$ 时，输出信号 $x$ 可以通过公式 6.4 计算出来：

$$x = \frac{V}{\mu}\left(e^{|y|\ln(1+\mu)/V} - 1\right)\operatorname{sgn}(y) \tag{6.4}$$

其中参数 $\mu$ 是 μ 律扩展参数，它与 μ 律压缩参数相等；$V$ 是输入信号的最大值，$\ln(x)$表示取 $x$ 的自然对数。μ 律扩展模块（Mu-Law Expander）如图 6-6 所示。

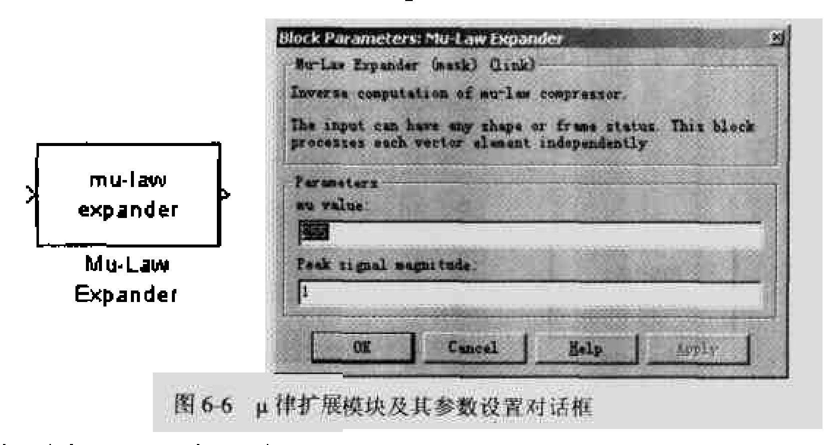



图 6-6 μ 律扩展模块及其参数设置对话框

μ 律扩展模块主要有以下两个参数。

■ mu value（参数 $\mu$）

$\mu$ 律扩展参数，对应于公式 6.4 中的参数 $\mu$。

■　Peak signal magnitude（输入信号的峰值）

输入信号的最大值，对应于公式 6.4 中的参数 $V$。

## 6.2　量化和编码

模拟信号经过抽样之后，其抽样值依然是随信号幅度连续变化的。当这些连续变化的抽样值通过噪声信道传输时，接收端很难准确地估计发送信号的大小。为此，抽样信号一般需要实施量化，把连续的抽样信号转换成离散的由有限个电平组成的信号。总的说来，量化是利用预先规定的有限个电平来表示模拟抽样值的过程。

编码是把信号的抽样量化值变换成代码的过程，其相反的过程称为译码。编码不仅用于通信，还广泛地应用于计算机、数字仪表和遥控遥测等领域。本节先介绍信源编码中的脉冲编码调制（PCM），关于信道编码的内容请参考第 7 章。

量化编码器用于把输入的连续信号转换成离散的数字信号。MATLAB 提供了两种类型的量化编码器：抽样量化编码器和触发式量化编码器，前者以恒定的速率产生抽样输出，后者则在触发信号的驱动下产生输出。量化解码器的作用与量化编码器相反，它把量化之后的信号还原为原始信号。本节将依次介绍这三种模块，以及量化编码器的一个使用实例。

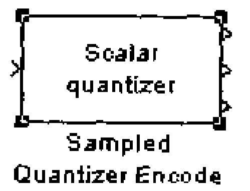
### 6.2.1　抽样量化编码器

抽样量化编码器（Sampled Quantizer Encode）根据量化间隔和量化码本把输入的模拟信号转换成数字信号，并且输出量化指标、量化电平以及误差的均方值。量化间隔是一个长度为 $n$ 的向量 $V$，其中的每个元素 $V(i)(i=1,2,\ldots,n)$严格单调递增。抽样量化编码器的量化指标 $y$ 是一个介于 0 和 $n+1$ 之间的整数，它由公式 6.5 确定。

$$y=\begin{cases}0, x\in(-\infty,V(1)]\\ m, x\in(V(m),V(m+1)], m\in[1,n-1]\\ n, x\in(V(n),+\infty)\end{cases}\tag{6.5}$$

量化码本是一个长度等于 $n+1$ 的向量 $C$，它把量化指标 $y$ 转换成输出信号的量化电平 $C(y+1)$，即当 $y$ 等于 0 时对应于量化码本的第一个元素 $C(1)$，当 $y$ 等于 $m$ 时对应于量化码本的第 $m+1$ 个元素 $C(m+1)$。

抽样量化编码器有三个输出端口。第一个输出端口输出量化指标 $y$，第二个输出端口输出信号的量化电平 $C(y)$，第三个输出端口则输出信号的量化误差。量化误差是根据输入信号与量化编码器的第二个输出端口的输出信号之差计算得到的均方值，它反映了量化编码器对信号的扭曲程度。抽样量化编码器模块及其参数设置对话框如图 6-7 所示。



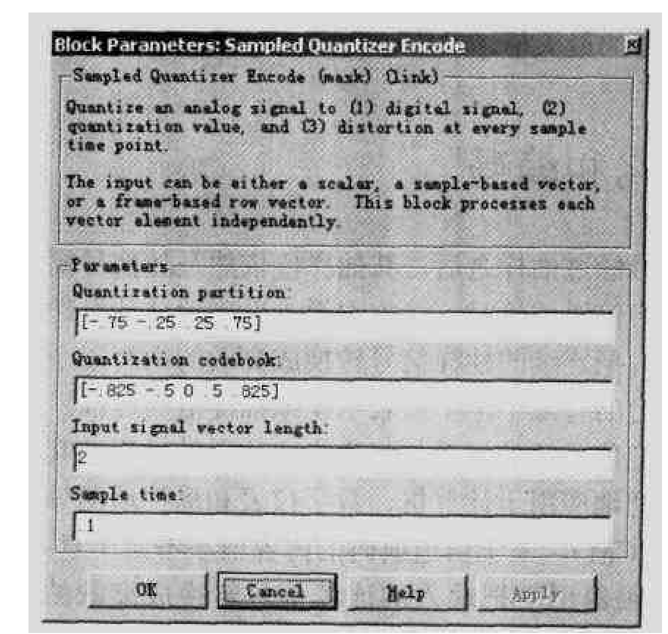




图 6-7　抽样量化编码器模块及其参数设置对话框

抽样量化编码器主要有以下几个参数：

- Quantization partition（量化间隔）

抽样量化编码器的间隔，它是一个长度为 $n$ 的向量 $V$，向量中的元素严格单调递增。

- Quantization codebook（量化码本）

抽样量化编码器的码本，它是一个长度为 $n$+1 的向量 $C$。

- Input signal vector length（输入信号向量长度）

当 Input signal vector length 等于 1 时，输入信号是一个标量；当 Input signal vector length 等于 $n$ 时，输入信号是一个 $n$ 维向量。

- Sample time（抽样时间）

输出信号的抽样时间间隔。

## 6.2.2　触发式量化编码器

触发式量化编码器（Enabled Quantizer Encode）类似于量化编码器，用于对输入信号进行量化编码，同时它还提供了另外一个输入端口，用于输入触发信号。当输入的触发信号不等于 0 时，触发式量化编码器产生一个新的输出；否则，维持原来的输出。由于触发式量化编码器只在触发信号有效的时候产生输出，因此它不需要设置抽样间隔。和量化编码器一样，触发式量化编码器的输入信号既可以是一个标量，也可以是一个向量。触发式量化编码器模块及其参数设置对话框如图 6-8 所示。

触发式量化编码器主要有量化间隔（Quantization partition）、量化码本（Quantization codebook）以及输入信号向量长度（Input signal vector length）3 个参数，参数的意义与 6.2.1 节中的参数相同。

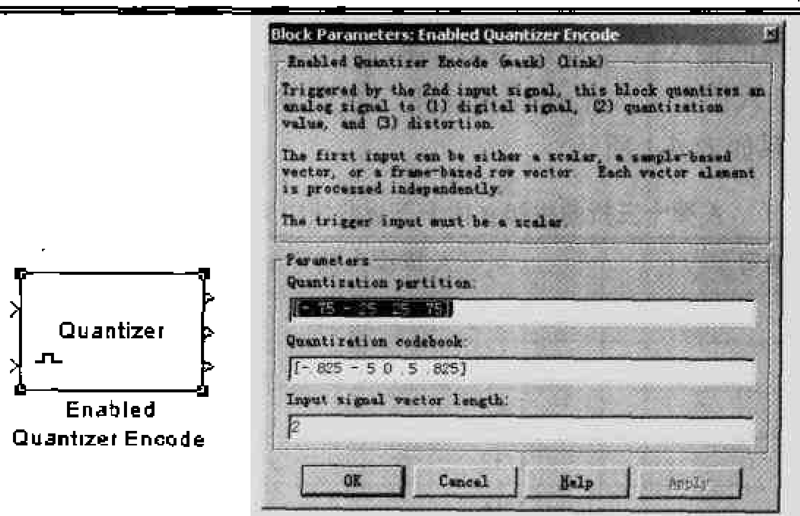



图 6-8　触发式量化编码器模块及其参数设置对话框

## 6.2.3　量化解码器

量化解码器（Quantizer Decode）是抽样量化编码器和触发式量化编码器的逆过程，它根据量化码本把量化编码器产生的量化指标转换成相应的电平。量化解码器的输出信号既可以是标量信号，也可以是向量信号。量化解码器模块及其参数设置对话框如图 6-9 所示。

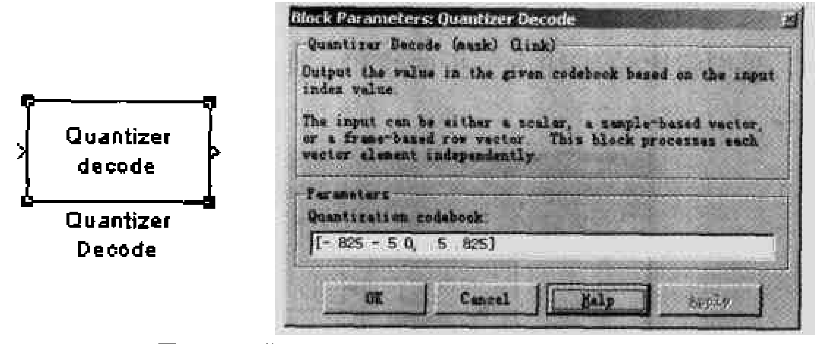



图 6-9　量化解码器模块及其参数设置对话框

量化解码器的量化码本应该与产生量化解码器输入信号的量化编码器适用的码本相同。实际上量化解码器的输入信号就是量化编码器的第一个输出端口的输出信号，量化解码器的输出信号等于量化编码器的第二个输出端口的输出信号。量化解码器只有一个参数。

■　Quantization codebook（量化码本）

量化解码器的码本，它是一个向量，并且它与相应的抽样量化编码器或触发式量化编码器使用的码本相同。

## 6.2.4　实例 6.1——A 律十三折与 μ 律十五折的量化误差

在 6.1 和 6.2 节里我们分别介绍了 A 律压缩和 μ 律压缩的原理及相应的模块，它们都可避免的带来一定的量化误差。同时，由于 A 律压缩和 μ 律压缩的计算公式比较复杂，难以在电路中实现，因此在实际应用中往往多采用近似于 A 律压缩函数和 μ 律压缩函数的曲线，由于这两种曲线分别有 13 折和 15 折，因此被分别称为 A 律十三折曲线和 μ 律十五折曲线。

设量化编码器的输入信号为 $x$，量化后的输出信号为 $y$，且 $x$ 和 $y$ 都是归一化信号，取值范围为[-1, 1]。输入信号 $x$ 大于 0 时 A 律十三折曲线和 μ 律十五折曲线中输入信号与输出信号之间的对应关系如表 6-1 所示。

**表6-1　　A 律十三折曲线和 μ 律十五折曲线中 $x$ 和 $y$ 的对应关系**

| 输出信号 $y$ | A 律输入信号 $x$ | μ 律输入信号 |
|---|---|---|
| 0 | 0 | 0 |
| 1/8 | 1/128 | 1/255 |
| 2/8 | 1/64 | 3/255 |
| 3/8 | 1/32 | 7/255 |
| 4/8 | 1/16 | 15/255 |
| 5/8 | 1/8 | 31/255 |
| 6/8 | 1/4 | 63/255 |
| 7/8 | 1/2 | 127/255 |
| 1 | 1 | 1 |

从表 6-1 所示中可以看出，当输入信号在-1/64 和 1/64 之间时，A 律十三折曲线的斜率恒等于 16，这样一来 A 律曲线就只有 13 个转折点了，这就是 A 律十三折曲线的由来；而 μ 律十五折曲线只在-1/255 和 1/255 之间斜率恒定，共有 14 个斜率发生变化的分界点，将其分成 15 段直折线。

在本实例中，我们将比较 A 律十三折曲线和 μ 律十五折曲线的量化误差，实例的模块框图如图 6-10 所示。

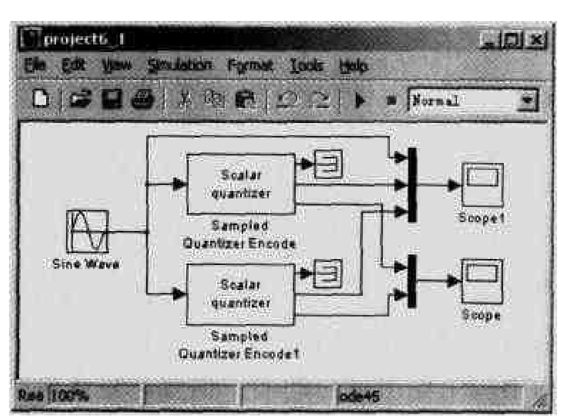



图 6-10　实例 6.1 的模块框图

在本实例中，我们采用一个 Sine Wave（正弦信号产生器）产生一个正弦信号，这个信号分别通过两个 Sampled Quantizer Encode 和 Sampled Quantizer Encode1（抽样量化编码器），按照 A 律十三折曲线和 μ 律十五折曲线产生量化输出信号，然后把这两个量化器计算得到的量化误差的均方值通过一个 Mux（复用器）输入到 Scope（示波器），这时候从示波器上就可以观察到这两种量化编码器产生的量化误差。为了比较量化之前和量化之后的正弦信号，正弦信号产生器和两个抽样量化编码器第二个输出端口的输出信号通过另外一个复用器连结到 Scope1（示波器）。

Sine Wave（正弦信号产生器）用来产生一个幅度为 1，频率为 1 赫兹的连续时间正弦信号，它的参数设置如表 6-2 所示。

表6-2　　正弦信号产生器的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Sine Wave |
| Sine type | Time based |
| Amplitude | 1 |
| Bias | 0 |
| Frequency (rad / sec) | 2*pi |
| Phase (rad) | 0 |
| Sample time | 0 |
| Interpret vector parameters as 1-D | Checked |

抽样量化编码器 Sampled Quantizer Encode 和 Sampled Quantizer Encode1 分别用于产生 A 律十三折曲线和 μ 律十五折曲线，它们把正弦信号产生器产生的正弦信号转换成量化信号，并且计算这个过程中产生的量化噪声。这两种抽样量化编码器的参数设置分别如表 6-3 和表 6-4 所示。

表6-3　　Sampled Quantizer Encode 的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Sampled Quantizer Encode |
| Quantization partition | [-1/2 -1/4 -1/8 -1/16 -1/32 -1/64 -1/128 0 1/128 1/64 1/32 1/16 1/8 1/4 1/2 1] |
| Quantization codebook | [-1 -7/8 -6/8 -5/8 -4/8 -3/8 -2/8 -1/8 0 1/8 2/8 3/8 4/8 5/8 6/8 7/8 1] |
| Input signal vector length | 1 |
| Sample time | 0.001 |

表6-4　　Sampled Quantizer Encode1 的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Sampled Quantizer Encode |
| Quantization partition | [-127/255 -63/255 -31/255 -15/255 -7/255 -3/255 -1/255 0 1/255 3/255 7/255 15/255 31/255 63/255 127/255 1] |
| Quantization codebook | [-1 -7/8 -6/8 -5/8 -4/8 -3/8 -2/8 -1/8 0 1/8 2/8 3/8 4/8 5/8 6/8 7/8 1] |
| Input signal vector length | 1 |
| Sample time | 0.001 |

图 6-11 所示是 Scope（示波器）的运行结果，其中黄线表示 Sampled Quantizer Encode（第一个抽样量化编码器）的量化误差的均方值，红线则表示 Sampled Quantizer Encode1（第二个抽样量化编码器）的量化误差。

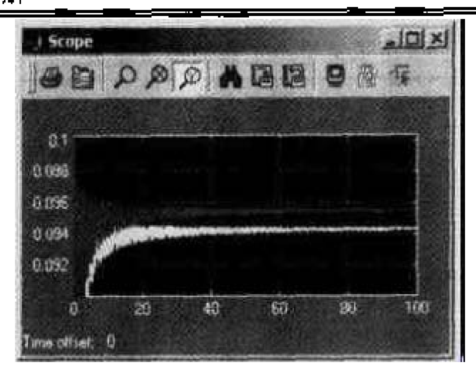
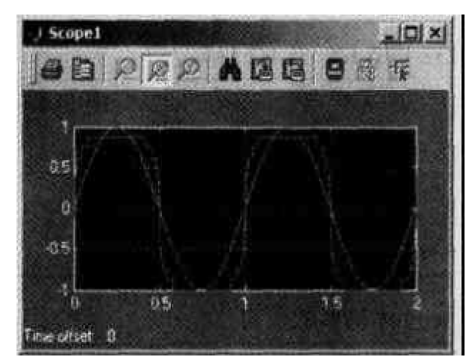



图 6-11 示波器 Scope 的运行结果

从图 6-11 所示中可以看出，当高斯噪声产生器的方差等于 0.01 时，按照 A 律十三折曲线进行量化编码产生的量化误差要比采用 μ 律十五折曲线是产生的量化误差小。事实上，μ 律十五折曲线在处理小信号的过程中能够得到更大的量化信噪比，而在处理大信号的过程中则性能要比 A 律十三折曲线差。

图 6-12 所示是 Scope1（示波器）的运行结果。图中的黄颜色线条表示抽样之前的正弦信号，青颜色的线条表示通过 Sampled Quantizer Encode（第一个抽样量化编码器）之后的信号，红颜色线条则表示通过 Sampled Quantizer Encode1（第二个抽样量化编码器）之后的信号。可以看到，抽样量化之后的信号与原来的连续信号之间存在着一定的量化误差，同时 A 律十三折曲线和 μ 律十五折曲线对大信号的处理方式相似，两者的差别在于对小信号的量化编码方式上，图 6-12 中的右图所示也说明了这一点。



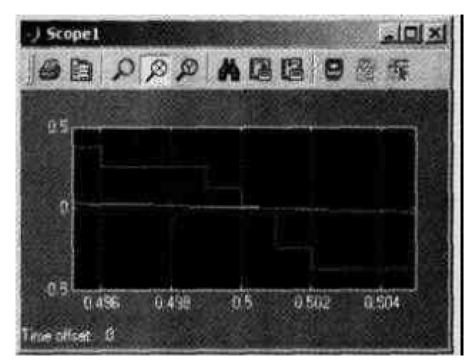




图 6-12 示波器 Scope1 的波形

# 6.3 差分编码

差分编码又称为增量编码，它用一个二进制位来表示前后两个抽样信号之间的大小关系。在 MATLAB 中，差分编码器根据当前时刻之前的所有输入信息计算输出信号，这样，在接收端就可以只按照接收到的前后两个二进制信号恢复出原来的信息序列。

## 6.3.1 差分编码器

差分编码器（Differential Encoder）的输入信号和输出信号都是二进制序列。假设输入信号为 $m(t)$，输出信号为 $d(t)$，则 $t_k$ 时刻的输出 $d(t_k)$ 不仅与当前时刻的输入信号 $m(t_k)$ 有关，而且与前一时刻的输出 $d(t_{k-1})$ 有关，如公式 6.6 所示。

$$\begin{cases} d(t_0) = (m(t_0) + 1) \bmod 2 \\ d(t_k) = (d(t_{k-1}) + m(t_k) + 1) \bmod 2 \end{cases} \tag{6.6}$$

将公式 6.6 叠代之后，可以得到等价的公式 6.7。

$$\begin{cases} d(t_0) = (m(t_0) + 1) \bmod 2 \\ d(t_k) = \left( \sum_{i=0}^{k} m(t_i) \right) \bmod 2, \text{当}k\text{是奇数时} \\ d(t_k) = \left( \sum_{i=0}^{k} m(t_i) + 1 \right) \bmod 2, \text{当}k\text{是偶数时} \end{cases} \tag{6.7}$$

即输出信号 $y$ 取决于当前时刻以及当前时刻之前的所有输入信号的数值。例如，假设输入信号 $m$ 等于 011001，则输出信号 $y$ 等于 111011。差分编码器模块及其参数设置对话框如图 6-13 所示。

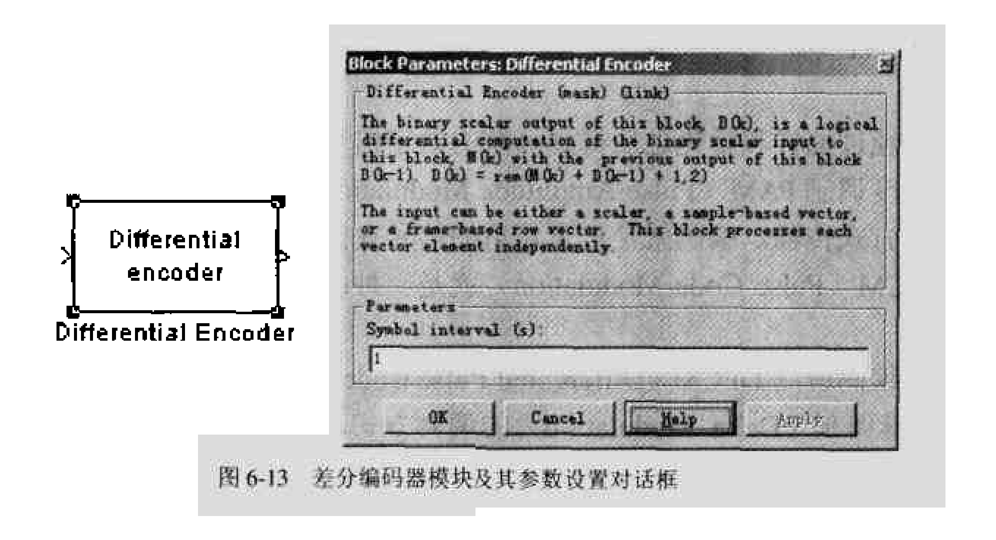



图 6-13　差分编码器模块及其参数设置对话框

差分编码器模块只有一个参数。

■ Symbol interval (s)（符号间隔）

输入信号符号之间的间隔。

## 6.3.2 差分解码器

差分解码器（Differential Decoder）的工作过程与差分编码器相反，它把差分编码器的输出信号还原成原始的二进制序列。假设差分解码器的输入信号序列为 $m$，输出信号序列为 $d$，则输入信号与输出信号的关系由公式 6.8 所示决定。

$$\begin{cases} d(t_0) = (m(t_0) + 1) \bmod 2 \\ d(t_k) = (m(t_{k-1}) + m(t_k) + 1) \bmod 2 \end{cases} \tag{6.8}$$

从公式 6.6 中可以看到，差分解码器的输出取决于离当前时刻最近的两个输入信号的数值。差分解码器模块及其参数设置对话框如图 6-14 所示。

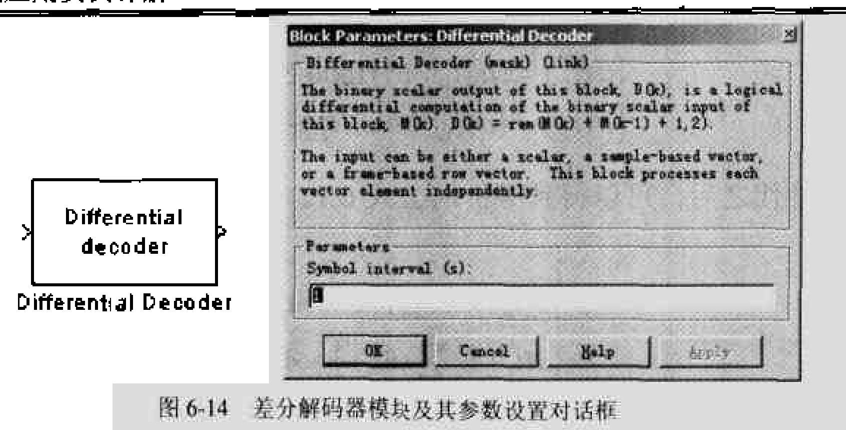



图 6-14　差分解码器模块及其参数设置对话框

差分解码器模块只有一个参数。

■　Symbol interval (s)（符号间隔）

输入信号符号之间的间隔。

## 6.4　DPCM 编码和解码

模拟信号经过抽样和量化之后得到离散的数字信号序列，这个数字信号序列可以通过 M 进制的脉冲幅度调制 PAM（Pulse Amplitude Modulation）直接进行传输，也可以把每一个量化电平用编码方式传输。编码就是把量化后的信号变换成代码，其相反的过程称为译码。脉冲编码调制 PCM（Pulse Code Modulation）就是一种编码方式，它把模拟信号的抽样量化值变换成代码。

差分脉冲编码调制 DPCM（Differential Pulse Code Modulation）是一种结合了脉冲编码调制 PCM 和增量调制（Delta Modulation）的调制方式，它首先把输入信号转换成增量信号，然后对这些信号进行 PCM 调制。图 6-15 所示是差分脉冲编码调制 DPCM 系统的组成。

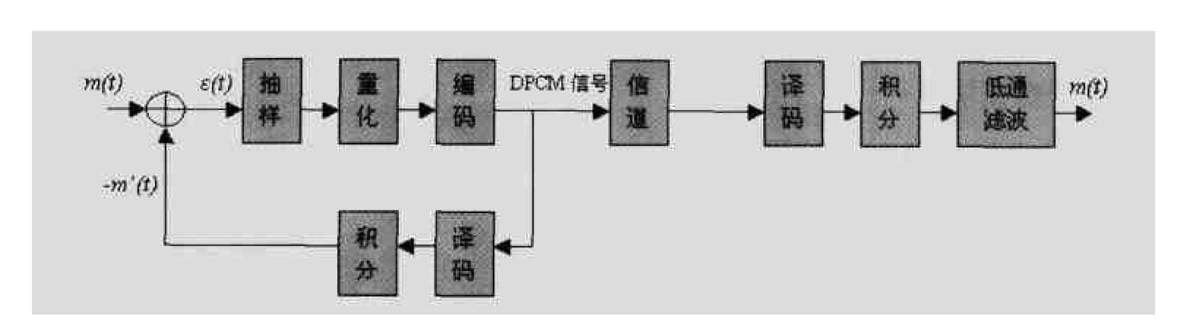



图 6-15　DPCM 系统的组成

### 6.4.1　DPCM 编码器

DPCM 编码器（DPCM Encoder）使用差分脉冲编码调制 DPCM（Differential Pulse Code Modulation）对输入信号进行量化和编码。DPCM 编码器的输入信号是一个标量，它产生两个输出信号：量化指标和量化编码信号。由于 DPCM 编码器内部使用了抽样量化编码器模块，因此它与抽样量化编码器一样有两个量化参数：量化间隔和量化码本。这两个参数都是一个

向量，其中量化间隔表示每个量化区间的端点，量化码本则表示每个量化区间的取值，并且量化码本向量的元素个数比量化区间的元素个数多 1。

我们在 6.1 节和 6.2 节中介绍的 A 律压缩和 μ 律压缩以及 6.3 节的量化编码器在当前时刻的输出只跟该时刻的输入信号有关，与先前的输入无关。实际应用中，通常可以根据先前的输入信号预测当前时刻的输出，这种方法称之为"预测量化"（predictive quantization）。脉冲差分调制 DPCM 就是一种预测量化编码。

假设在时刻 $k$ 的输入信号为 $x(k)$，输出信号为 $y(k)$，则预测量化的公式如公式 6.9 所示。

$$y(k)=p(1)x(k-1)+p(2)x(k-2)+\cdots+p(m-1)x(k-m+1)+p(m)x(k-m) \qquad (6.9)$$

其中 $p$ 表示预测量化的系数，它是一个 $m$ 维的实向量，$m$ 称为预测量化的维数。预测量化多项式（公式 6.7）还可以表示成向量的形式[0, $p(1)$, $p(2)$, $p(3)$,..., $p(m\text{-}1)$, $p(m)$]，其中第一个元素 0 不能忽略，因为 MATLAB 把预测量化看作是一个有限冲击响应 FIR（Finite Impulse Response）滤波器。当 $m$ 等于 1 时，$y(k)=p(1)x(k-1)$，它表示的是增量调制（Delta Modulation）。

为了在 DPCM 编码器中实现预测量化，除了需要指定量化间隔和量化码本之外，还需要确定预测量化多项式。DPCM 编码器模块及其参数设置对话框如图 6-16 所示。

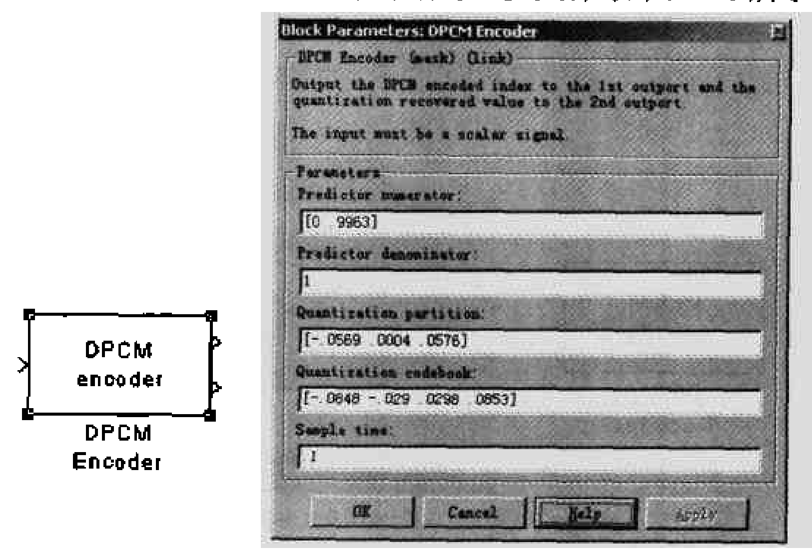



图 6-16　DPCM 编码器模块及其参数设置对话框

DPCM 编码器模块主要有以下几个参数。

■　Predictor numerator（预测多项式分子）

由预测多项式（公式 6.9）中每个系数 $p(i)$的分子组成的向量，该向量的第一个元素必须等于 0。

■　Predictor denominator（预测多项式分母）

由预测多项式（公式 6.9）中每个系数 $p(i)$的分母组成的向量。预测多项式分母 Predictor denominator 通常设置为 1。

■　Quantization partition（量化间隔）

DPCM 编码器的量化间隔，它是一个 $n$ 维向量，并且把输入信号的取值范围分成 $n$+1 个量化区间。

- Quantization codebook（量化码本）

DPCM 编码器的量化码本，它是一个 $n+1$ 维向量，其中的每个元素对应于一个量化区间的取值。

- Sample time（抽样时间）

DPCM 编码器的抽样时间。

## 6.4.2 DPCM 解码器

DPCM 解码器（DPCM Decoder）是用于还原 DPCM 信号的模块，它的输入信号是一个标量，表示 DPCM 信号的量化指标（即表示信号处于第几个量化间隔）。DPCM 解码器有两个输出端口，第一个输出端口输出还原之后的信号，第二个输出端口则表示根据预测量化算法得到的输出，它等价于 DPCM 编码器第二个端口的输出。DPCM 解码器模块及其参数设置对话框如图 6-17 所示。

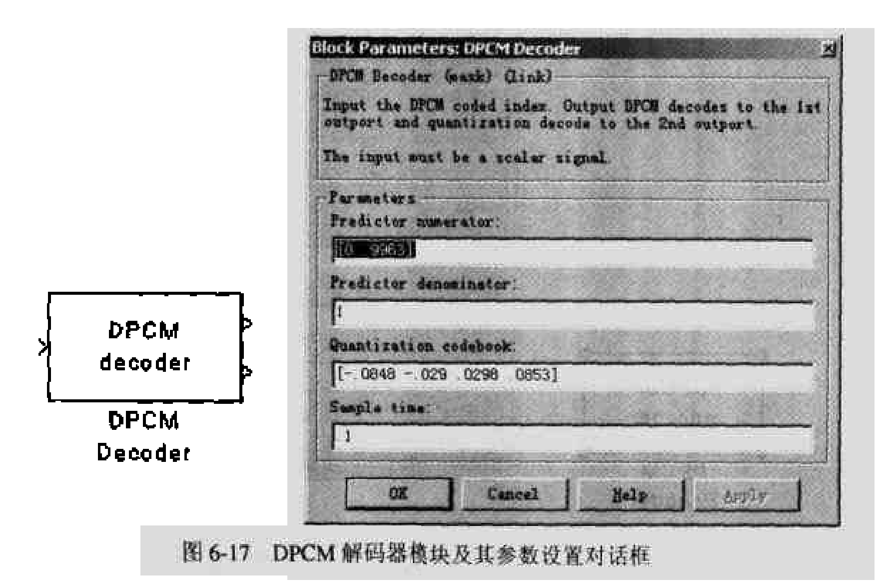



图 6-17 DPCM 解码器模块及其参数设置对话框

和 DPCM 编码器一样，DPCM 解码器采用预测量化的方法估计输出值，并且计算出这个过程中产生的误差。DPCM 解码器的参数应该与相应的 DPCM 编码器的参数保持一致。DPCM 解码器模块主要有 Predictor numerator（预测多项式分子）、Predictor denominator（预测多项式分母）、Quantization codebook（量化码本）和 Sample time（抽样时间）4 个参数，参数的意义与 6.4.1 中的参数相似。

## 6.4.3 实例 6.2——DPCM 与 PCM 系统的量化噪声

在通信系统中，信源编码是把量化后的信号变换成代码的过程，其相反的过程称为译码。编码不仅用于通信，还用于计算机、数字仪表以及遥控测控等领域。

在 A 律十三折线或 μ 律十五折线 PCM 编码中，一般采用八位二进制编码，并且这八位二进制码分成三部分：极性码、段落码和段内码。极性码表示量化值的极性，它占一个二进

制位；第二位至第四位表示段落码，它是按照 A 律十三折线或 μ 律十五折线对信号的绝对值进行不均匀量化的量化值；第五位至第八位是段内码，它在 A 律十三折线或 μ 律十五折线不均匀量化的基础上对信号实施均匀量化，将相应的区间分割成 16 个相等的区间，段内码就是这些均匀区间的编号。

在本实例中，我们分别采用抽样量化编码器和 DPCM 编码器来产生 PCM 信号和 DPCM 信号，为此我们需要首先确定抽样量化编码器和 DPCM 编码器的量化码本及其量化间隔。下面的程序段由于产生 A 律十三折线 PCM 编码的量化间隔 partition 和量化码本 codebook，它们分别具有 256 和 257 个元素。

```
>>format rat
>>%设置工作区的显示方式，把浮点数表示成分数
>>
>>sample=0.01;
>>%设置抽样时间，在后边的仿真过程中需要要到这个参数
>>
>>t=[1:16]/16;
>>%用 t 作为临时变量
>>
>>x=[t/128 (1+t)/128 (1+t)/64 (1+t)/32 (1+t)/16 (1+t)/8 (1+t)/4 (1+t)/2];
>>%x 是单边的量化间隔
>>
>>for i=1:128
>>y(i)=x(129-i);
>>end
>>y=-y;
>>%构造另一边的量化间隔
>>
>>codebook=[y 0 x];
>>%构造量化码本
>>
>>partition=codebook(1:256);
>>%构造量化间隔
```

本实例将使用抽样量化编码器和 DPCM 编码器分别对同一个正弦信号进行量化和编码，产生八位的脉冲编码信号。与编码器对应的解码器对量化信号实施解码，然后统计解码信号与原始的正弦信号之间的均方差，由此得到 DPCM 和 PCM 两种方式下的量化误差。整个实例的模块结构图如图 6-18 所示。

本系统的信号源是一个 Sine Wave（正弦信号产生器），它用来产生一个幅度为 1，频率为 1 赫兹的连续时间正弦信号。正弦信号产生器的参数设置如表 6-5 所示。

表6-7　Sampled Quantizer Encode 的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Sampled Quantizer Encode |
| Quantization partition | partition |
| Quantization codebook | codebook |
| Input signal vector length | 1 |
| Sample time | sample |

表6-8　DPCM decoder 的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | DPCM Decoder |
| Predictor numerator | [0 1] |
| Predictor denominator | 1 |
| Quantization codebook | codebook |
| Sample time | sample |

表6-9　Quantizer decode 的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Sampled Quantizer Decode |
| Quantization codebook | codebook |

在前面的程序段中，我们在 MATLAB 工作区设置了 3 个变量 codebook、partition 以及 sample，所以在设置模块的参数时可以直接输入这些变量的名称，这样就简化了参数的重复输入过程，同时也便于修改参数的数值。

在图 6-18 所示的模块框图中，我们使用了一个子系统 Performance，专门用于观察本实例的运行结果。它从正弦信号产生器和两个量化解码器得到输入信号，对这些信号作相应的变换之后在示波器上显示出来。该子模块如图 6-19 所示。

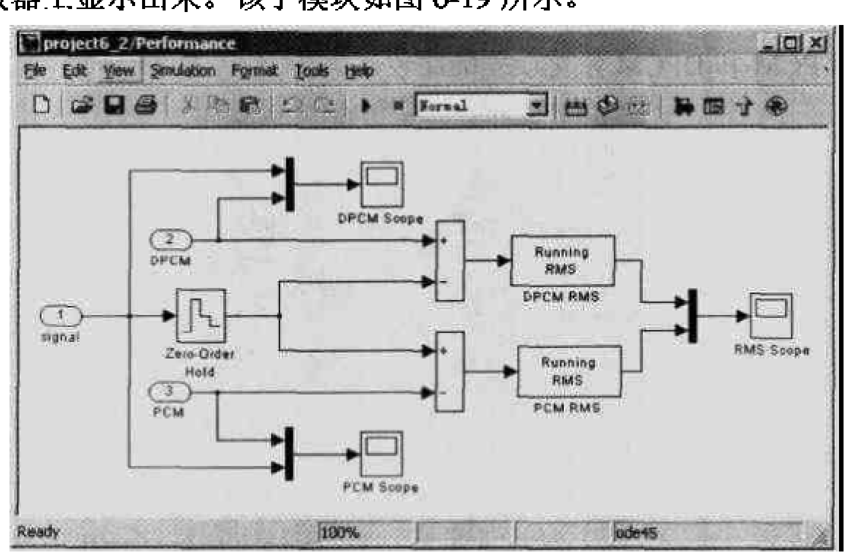
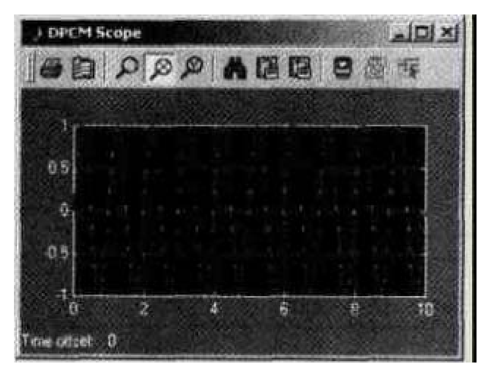



图 6-19　Performance 子系统的模块结构框图

在 Performance 子系统中，DPCM 解码器和抽样量化解码器输出的解码信号分别在示波

器 DPCM Scope 和示波器 PCM Scope 上显示出来。作为一个对比，这两个示波器同时显示了正弦信号产生器输出的正弦信号，如图 6-20 所示。



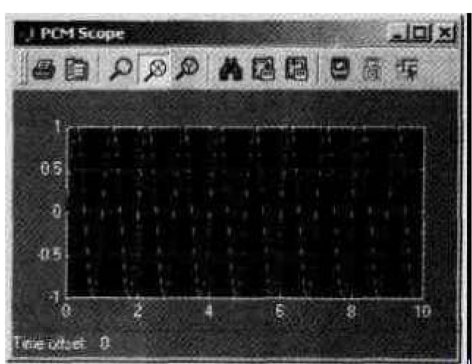




图 6-20　PCM 信号和 DPCM 信号解码后的波形与原信号波形的比较

从图 6-20 所示中可以看到，解码之后的信号与原来的正弦信号是相当接近的，而且 PCM 和 DPCM 这两种调制方式都能够很好地恢复出原始信号。Performance 子系统还分别计算 PCM 和 DPCM 两种调制方式产生的量化误差。Performance 子系统比较原始的正弦信号以及解码之后的信号之间的误差，并且计算这种误差的均方值。均方值的计算是通过两个 RMS 模块（即 DPCM RMS 和 PCM RMS）实现的，这两个 RMS 模块的参数设置相同，如表 6-10 所示。

**表6-10　RMS 模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | RMS |
| Running RMS | Checked |
| Reset port | None |

由于正弦信号产生器输出的正弦信号是一个连续时间信号，因此在执行减法运算之间必须把它转换成抽样信号。图 6-19 所示中的 Zero Order Hold 模块就是为此目的而设置的，在这个模块中，它的抽样时间 Sample time 设置为 sample/10。通过 DPCM RMS 模块和 PCM RMS 模块计算得到的量化误差输出到示波器 RMS Scope 中，用来比较两种编码调制方式的性能。图 6-21 所示是对 PCM 和 DPCM 量化误差的比较。

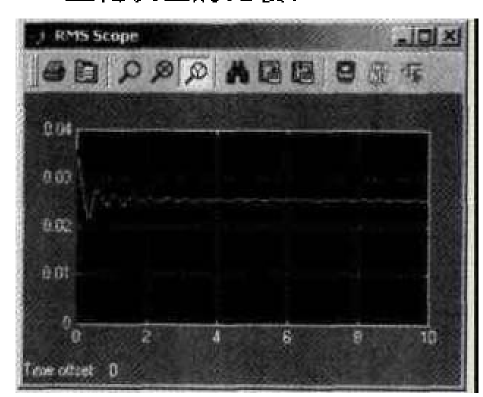



图 6-21　PCM 和 DPCM 量化误差的比较

在图 6-21 所示中，黄颜色绘制的线条表示的是 DPCM 信号的量化误差，红颜色线条则表示 PCM 信号的量化误差。从图中可以清楚地看到，DPCM 信号的量化误差低于 PCM 信号的量化误差，这从一个侧面说明了 DPCM 编码调制方式的优势。

最后我们再来看一看通过 DPCM 编码器以及抽样量化编码器之后产生的 PCM 信号和 DPCM 信号，如图 6-22 所示，其中第一个图表示 DPCM 编码信号，第二个图表示 PCM 编码信号。

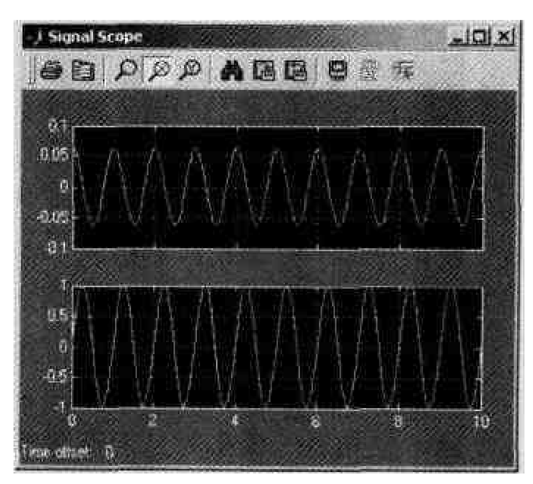



图 6-22 DPCM 信号和 PCM 信号的波形

从图 6-22 所示中可以看出，DPCM 编码信号的也是一个正弦信号，但是幅度值只有大约 0.05。与此形成鲜明对比的是，PCM 信号的幅度值是 1。造成这种现象的原因在于，DPCM 先计算输入信号的差值，然后再对这个差值实施 PCM 编码。假设输入信号为 $\sin(2\pi t)$，抽样间隔为 $t_s$，则对于相邻的两个时刻 $t$ 和 $t+t_s$，它们的信号差值等于 $\sin(2\pi(t+t_s))-\sin(2\pi t)$，经过三角变化之后得到 $2\sin(\pi t_s)\cos(2\pi t+\pi t_s)$。因此，正弦信号的差分信号依然是一个正弦型信号，但是其幅度变成了 $2\sin(\pi t_s)$。具体到我们这个实例程序，由于抽样时间 $t_s$ 等于 0.01，计算得到 $2\sin(\pi t_s)=0.0628$，这就是 DPCM 信号幅度变小的原因。

# 第7章　信道编码和交织

在移动通信系统中，发送端发出的无线信号不可避免地要受到随机噪声和突发噪声的影响，信号的传输波形若受到破坏，则会使得接收端可能发生错误判决。消除或降低噪声干扰的方法包括采用均衡器，提高信号的发射功率等等，但是这些方法通常不能满足传输要求。

信道编码是现代通信系统广泛采用的一种差错控制措施。在信道编码过程中，发送端通过某种方式对信息序列进行计算，得到相应的检错/纠错编码，然后把这个检错/纠错编码附加到信息序列中，经过载波调制之后发射出去。接收端对接收到的信号进行解调，从中恢复出包含原始信息序列和检错/纠错编码的二进制序列。对这个序列实施信道编码的逆过程之后，就得到了所需的信息序列，同时也知道了该序列是否存在着传输错误。如果信道编码只具有检错能力，它可以通过发送端的重传过程来纠正这个错误；如果信道编码还具有一定的纠错能力，它就有可能纠正这个错误，但是前提是接收信号中误码的个数没有超出信道编码的纠错能力。

信道编码可以分成分组码、卷积码和循环冗余码3类（严格地说，循环冗余码属于信源编码的范畴，把它放在本章里是为了论述的方便）。本章将依次介绍这3种编码的原理和使用方法。

## 7.1　分组编码

分组编码是把若干个输入信号变换成一个更长的输出序列的编码方式，它通过提供编码的冗余度来实现对信号的检错和纠错。假设输入信号是一个长度为 $k$ 的向量 $x$，经过分组编码之后的输出信号是一个长度为 $n$ 的向量 $y$，则这个分组编码表示为（$n$, $k$），其中信息位的长度等于 $k$，码长为 $n$，监督位的长度 $r=n-k$，编码效率等于 $k/n=1-r/k$。对于分组码，输出序列 $y$ 一般可以表示成输入向量 $x$ 与生成矩阵 $G$ 的乘积，即 $y=xG$，其中 $G$ 是一个 $k$ 行 $n$ 列的矩阵。

### 7.1.1　二进制线性码

线性码是一种分组编码，在这种编码中，信息位和监督位是由一些线性代数方程联系着的，或者说，线性码是按一组线性方程构成的。线性分组码的范围较广，后面介绍的循环码、BCH 码、Reed-Solomon 码和 Hamming 码都可以看作是线性码的特例。

**1．二进制线性编码器**

二进制线性编码器（Binary Linear Encoder）根据生成矩阵 $G$ 产生二进制线性码。假设输入信号是一个长度为 $k$ 的行向量 $v=(v_1,v_2,\cdots v_k)$，则生成矩阵 $G$ 是一个 $k$ 行 $n$ 列的矩阵 $G=(g_{ij})_{k\times n}$，并且二进制线性编码器的输出信号等于：

$$vG=\left(\sum_{i=1}^{k}v_ig_{i1},\sum_{i=1}^{k}v_ig_{i2},\cdots,\sum_{i=1}^{k}v_ig_{in}\right) \tag{7.1}$$

因此，二进制线性编码器的输出是一个长度为 $n$ 的行向量。二进制线性编码器模块及其参数设置对话框如图 7-1 所示。

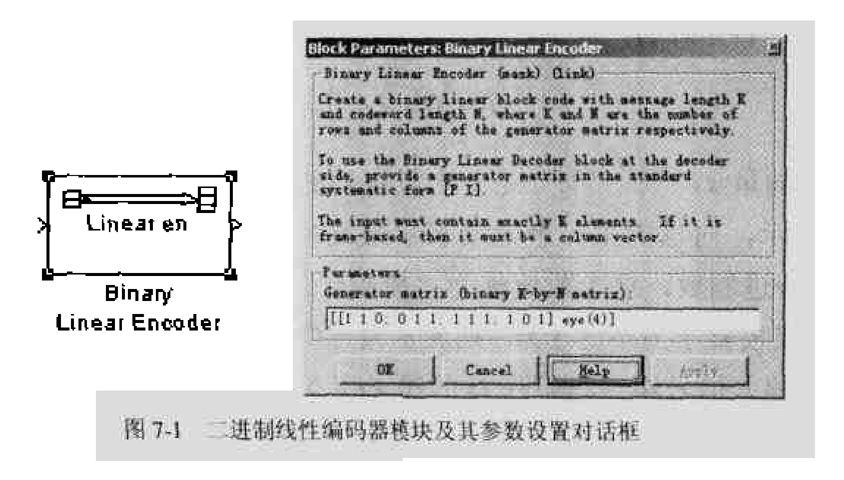



图 7-1 二进制线性编码器模块及其参数设置对话框

二进制线性编码器只有一个参数。

■ Generator matrix（生成矩阵）

生成矩阵 $G$，它是一个 $k$ 行 $n$ 列的矩阵。

## 2．二进制线性解码器

二进制线性解码器（Binary Linear Decoder）根据生成矩阵对二进制线性编码器的编码信号进行解调，得到原始的二进制信号序列。二进制线性解码器的生成矩阵应该与二进制线性编码器的生成矩阵保持一致。如果生成矩阵 $G$ 是一个 $k$ 行 $n$ 列的矩阵，则二进制线性解码器的输入信号 $X$ 是一个 $n$ 列的行向量（或矩阵），并且产生一个 $k$ 列的行向量（或矩阵）$Y$。二进制线性解码器模块及其参数设置对话框如图 7-2 所示。

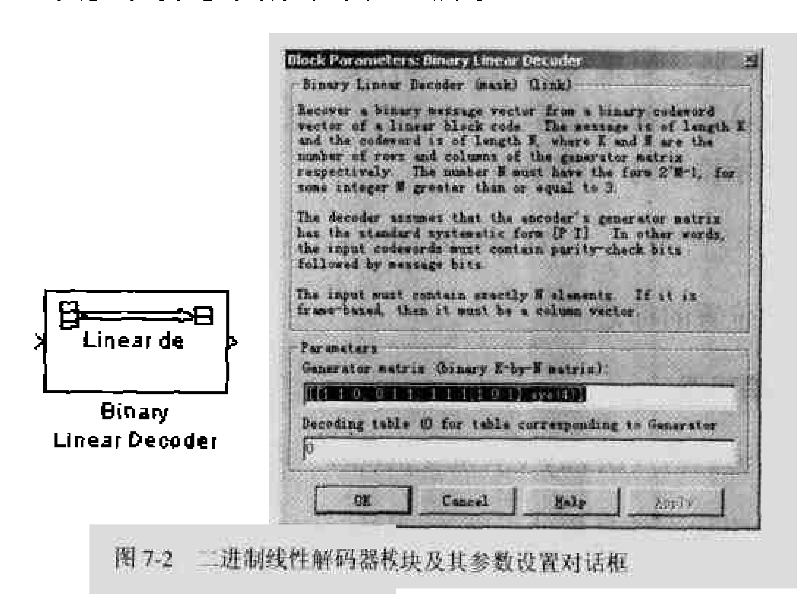



图 7-2 二进制线性解码器模块及其参数设置对话框

二进制线性解码器有一个译码表（Decoding table），用于对输入信号进行纠错。译

码表是一个$2^{n-k}$行$n$列的矩阵$D$，其中的每一个行向量都对应于一个纠正后的编码。对于生成矩阵$G$，如果它可以表示成一个$k$阶单位矩阵$I_k$与一个$k$行$n$-$k$列矩阵$P_{k\times(n-k)}$的组合：

$$G=[I_k,Q_{k\times(n-k)}] \tag{7.2}$$

根据$G$构造监督矩阵$H$：

$$H=[Q^T,I_{n-k}] \tag{7.3}$$

则对于输入向量$x$，$xH^T=Hx^T$是一个$n$-$k$列的向量。如果把这个向量看作是一个十进制数的二进制表示，向量的第一个元素表示最高位，则可以把向量表示成一个介于 0 和$2^{n-k}-1$之间的十进制数，这个数值称为“校验子”。如果校验子等于零，则输入信号没有差错；否则，如果校验子$x$不等于 0，则表示输入信号在传输过程中出现错误，这时候二进制线性解码器把输入信号纠正为矩阵$D$的第$x$+1 个行向量。

二进制线性解码器模块有两个参数。

■ Generator matrix（生成矩阵）

生成矩阵$G$，它是一个$k$行$n$列的矩阵，并且应该与相应的二进制线性编码器的生成矩阵相同。

■ Decoding table（译码表）

译码表是一个$2^{n-k}$行$n$列的矩阵。当译码表 Decoding table 被设置为 0 时，由 MATLAB 自动生成一个译码表。

### 7.1.2 二进制循环码

循环码除了具有线性码的一般性质外，还具有循环性，即循环码中任意码组循环移动一位（将最右端的码元移至左端，或把最左端的码元移至右端）之后，仍然是这个循环码中的一个码组。一般说来，如果$(a_{n-1}a_{n-2}\cdots a_1)$是一个循环码组，则$(a_{n-2}a_{n-3}\cdots a_1a_{n-1})$，$(a_{n-3}a_{n-4}\cdots a_{n-1}a_{n-2})$等等都是该循环码的码组。在代数编码理论中，为了便于计算，通常用码多项式$T(x)$来表示循环码中的码组：

$$T(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0 \tag{7.4}$$

其中的$x$只是码元位置的标记。

#### 1. 二进制循环码编码器

二进制循环码编码器（Binary Cyclic Encoder）用来产生一个（$n, k$）系统循环码（即信息位不变，监督位附加在信息位的后面），其中$n=2^m-1, m\ge 3$。二进制循环码编码器模块的输入信号是一个 k 维行向量，输出信号是$n$维行向量。二进制循环码编码器模块及其参数设置对话框如图 7-3 所示。

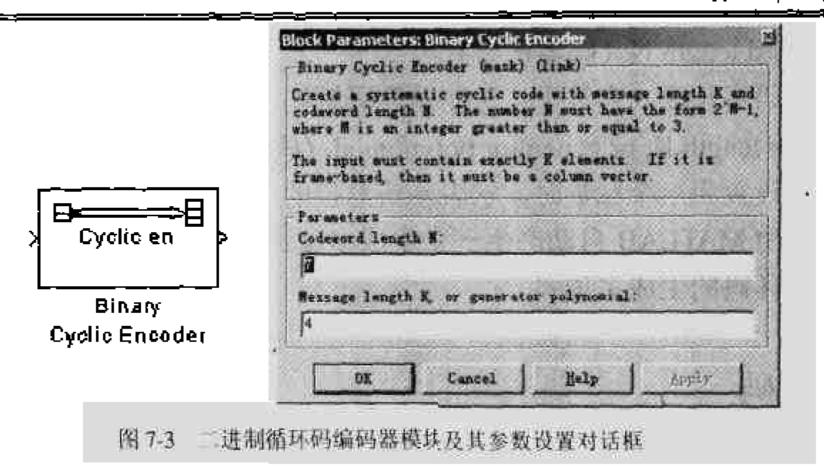



图 7-3　二进制循环码编码器模块及其参数设置对话框

二进制循环码编码器模块可以由用户自己指定一个生成多项式，也可以由 MATLAB 自动产生一个生成多项式，这是通过调用 MATLAB 函数 cyclpoly(*n*, *k*, 'min')实现的。

二进制循环码编码器模块有两个参数。

■ Codeword length N（码字长度）

循环编码的码字长度，它等于二进制循环码编码器产生的输出信号的长度。

■ Message length K, or generator polynomial（信息位长度/生成多项式）

当本参数被设置为一个标量时，它表示信息位的长度，即二进制循环码编码器输入信号的长度，这时候由 MATLAB 自动产生一个生成多项式；当本参数设置为一个二进制向量时，它表示二进制循环码的生成多项式。

## 2．二进制循环码解码器

二进制循环码解码器（Binary Cyclic Decoder）用于对二进制系统循环码进行解码，它的输入信号是一个长度为 $n$ 的向量，其中 $n=2^m-1, m\geq 3$，根据生成多项式还原得到长度为 $k$ 的二进制序列。二进制循环码解码器模块及其参数设置对话框如图 7-4 所示。

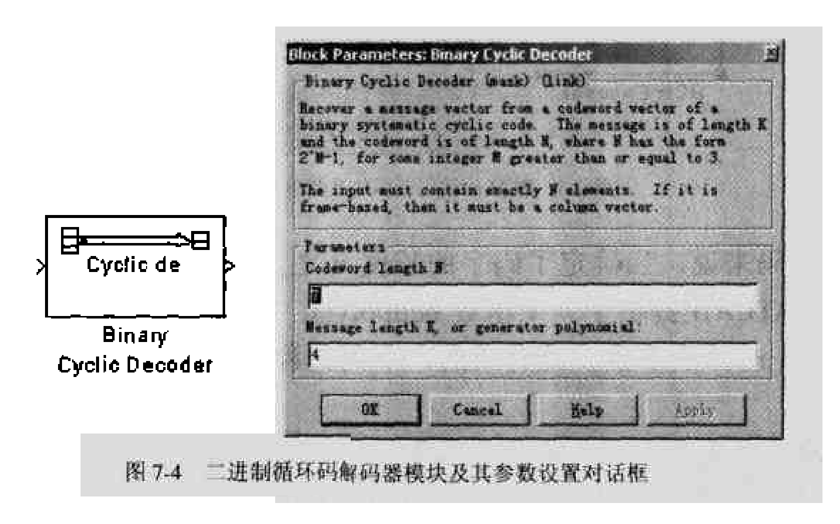



图 7-4　二进制循环码解码器模块及其参数设置对话框

与二进制循环码编码器模块一样，二进制循环码解码器模块中的生成多项式既可以由用户自己指定，也可以由 MATLAB 自动产生。值得注意的是，解码器中的生成多项式应该与编码器中的生成多项式保持一致。二进制循环码解码器模块有两个参数。

- Codeword length N（码字长度）

循环编码的码字长度，它等于二进制循环码解码器输入信号的长度。

- Message length K, or generator polynomial（信息位长度/生成多项式）

当本参数被设置为一个标量时，它表示信息位的长度，即二进制循环码解码器输出信号的长度，这时候由 MATLAB 自动产生一个生成多项式；当本参数设置为一个二进制向量时，它表示二进制循环码的生成多项式。

## 7.1.3 BCH 码

BCH 码是一种特别重要的循环码，它是根据 3 个发明人的名字 Bose、Chaudhuri 和 Hocguenghem 命名的。BCH 码的重要性在于它解决了生成多项式与纠错能力的关系问题，可以方便地纠正多个随机错误。对于特定的码字长度 $n$，BCH 码只能对特定的长度为 $k$ 的信息序列进行编码。

### 1．BCH 码编码器

BCH 码编码器（BCH Encoder）把 $k$ 位信息序列转换成 $n$ 位编码序列，它的输入信号包含 $k$ 个元素，输出信号是一个长度为 $n$ 的向量，其中 $n=2^m-1, m\geq 3$。BCH 码编码器模块及其参数设置对话框如图 7-5 所示。

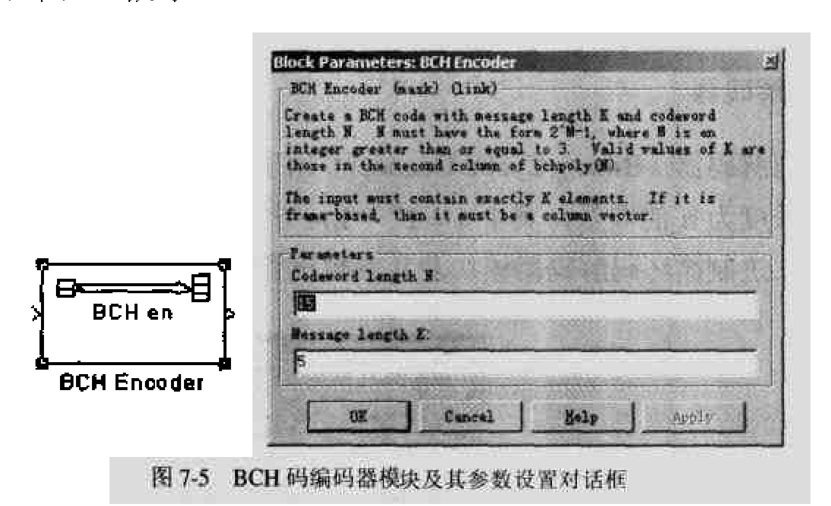



图 7-5　BCH 码编码器模块及其参数设置对话框

对于 BCH 码来说，当确定了码字的长度 $n$ 之后，只有特定信息序列长度 $k$ 才能产生 BCH 码。MATLAB 提供了一个函数 bchpoly()，用于验证当 $n$ 等于 7、15、31、63、127、255 或 511 时哪些参数 $k$ 是有效的。下列的程序段列出了 $n$ 等于 15 时所有的 $k$ 的数值。

```
<<params = bchpoly(15)
params =
    15    11    1
    15     7    2
    15     5    3
```

在上面的程序段中，第一列参数表示码字的长度 $n$，第二列的参数表示允许的信息位的长度 $k$，第三列的参数则表示这种编码的纠错能力。

BCH 码编码器模块有两个参数。

■ Codeword length N（码字长度）

BCH 码的码字长度，它等于 BCH 码编码器模块的输出向量的长度。

■ Message length K（信息位长度）

BCH 码的信息位长度，它等于 BCH 码编码器模块的输入向量的长度。

2. BCH 码解码器

BCH 码解码器（BCH Decoder）用于对 BCH 码序列进行解码，得到原始的信息序列。如果 BCH 编码的信息位长度为 $k$，编码后的码字长度为 $n$，则 BCH 码解码器的输入信号是一个长度为 $n$ 的向量，并且第一个输出端口的输出向量的长度为 $k$，其中 $n = 2^m - 1, m \geq 3$，$k$ 是符合函数 bchpoly(n)的一个输出的数值。BCH 码解码器模块及其参数设置对话框如图 7-6 所示。

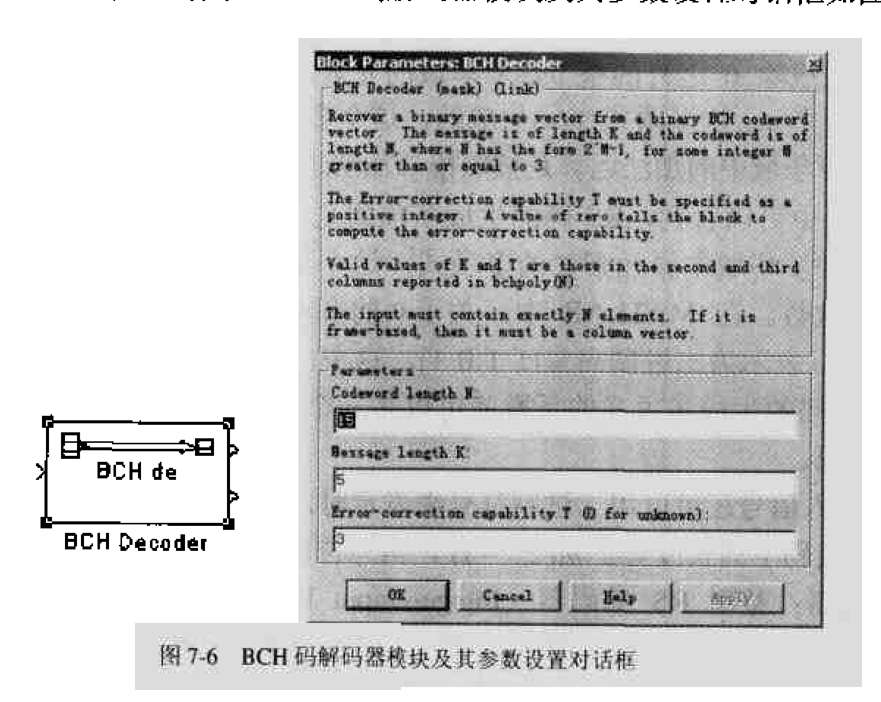



图 7-6　BCH 码解码器模块及其参数设置对话框

BCH 码解码器模块的第二个输出端口是在解码过程中检测到的错误符号的个数。当第二个输出端口的数值是一个负整数时，表示误码的个数超过了 BCH 码解码器的纠错能力 $T$。BCH 码解码器模块有 3 个参数。

■ Codeword length N（码字长度）

BCH 码的码字长度，它等于 BCH 码解码器模块的输入向量的长度。

■ Message length K（信息位长度）

BCH 码的信息位长度，它等于 BCH 码解码器模块第一个输出端口的输出向量的长度。

■ Error-correction capability T（纠错能力）

表示 BCH 码解码器的纠错能力。当本参数等于 0 时，MATLAB 自动计算 BCH 码的纠错能力。如果用户知道输入的 BCH 码信号的纠错能力，可以手工设置 Error-correction capability

T 为相应的正整数。

### 7.1.4 Reed-Solomon 码

Reed-Solomon 码是根据这种码的两个发明人的名字 Reed 和 Solomon 命名的，它是一种具有很强纠错能力的多进制 BCH 码，简称 RS 码。对于一个 M 进制 RS 码，它的输入信号和输出信号的取值范围都等于$[0, M-1]$，其中$M=2^m$。RS 码的码字长度 n 等于$n=M-1=2^m-1$，如果信息位的长度等于 k，则监督位的长度$r=n-k$。另外，RS 码具有很强的纠错能力，假设它能够纠正 t 个错，则 RS 码的监督位长度 r 和 t 之间应该满足关系$r=n-k=2t$，因此，RS 码的码字长度与信息位长度之间的差值应该是一个偶数，同时，RS 的最小码元距离$d_0=r+1=2t+1$。

RS 码的生成多项式 $g(x)$具有如下形式：

$$g(x)=(x+\alpha)(x+\alpha^2)\cdots(x+\alpha^{2t}) \tag{7.5}$$

式中的$\alpha$是伽罗华域（Galois field）$GF(2^m)=\{0,1,\alpha,\alpha^2,\cdots,\alpha^{2^m-2}\}$中的本原元素（本原元素的各次幂能够生成除 0 之外的整个伽罗华域的元素，其中 α 就是一个本原元素）。伽罗华域$GF(2^m)$是由一个 $m$ 次多项式 $p(x)$生成的，这个多项式称为本原多项式，并且 α 满足条件$p(\alpha)=0$。伽罗华域中的加法运算是按照模二加法进行的。

从公式 7.5 中可以看到，RS 码的生成多项式是一个 r 次多项式，其中每个项 x 的系数都是一个关于伽罗华域本原元素$\alpha$的多项式，这个多项式的最高幂次小于这个伽罗华域的本原多项式的次数。在 MATLAB 中，关于 α 的多项式可以表示成一个十进制数，例如，$\alpha^3+\alpha^2+1$可以表示为二进制向量[1 1 0 1]，这个二进制向量对应的十进制数是 13。通过这种方式，RS 码的生成多项式的系数就可以是一个多进制整数，从而 RS 码能够实现对多进制整数的编码。

RS 码的输入信号还可以用二进制符号来表示时，每个 M 进制符号可以表示为 m 位二进制符号。这时候码字的长度等于$m(2^m-1)$，信息位的长度等于$m\times k$。MATLAB 提供了两种 RS 码编码器：整型 RS 编码器（Integer-Input RS Encoder）以及二进制 RS 码编码器（Binary-Input RS Encoder）。相应地，MATLAB 中有两种 RS 码解码器：整型 RS 解码器（Integer-Input RS Decoder）以及二进制 RS 码解码器（Binary-Input RS Decoder）。在后面的几个小节里，我们将依次介绍这 4 种编码解码器。

#### 1. 整型 RS 码编码器

整型 RS 编码器产生一个信息位长度为 k，编码长度为 n 的 Reed-Solomon 码。值得注意的是，整型 RS 编码器的输入信号只能是帧信号，并且每帧的长度等于 k 的整数倍，帧中每个元素的取值范围是$[0,2^m-1]$，其中$m=\lceil\log_2(n+1)\rceil$。经过编码之后的输出信号也是一个帧信号，每帧的长度是 n 的整数倍，每个元素的取值范围也是$[0,2^m-1]$。整型 RS 编码器模块及其参数设置对话框如图 7-7 所示。

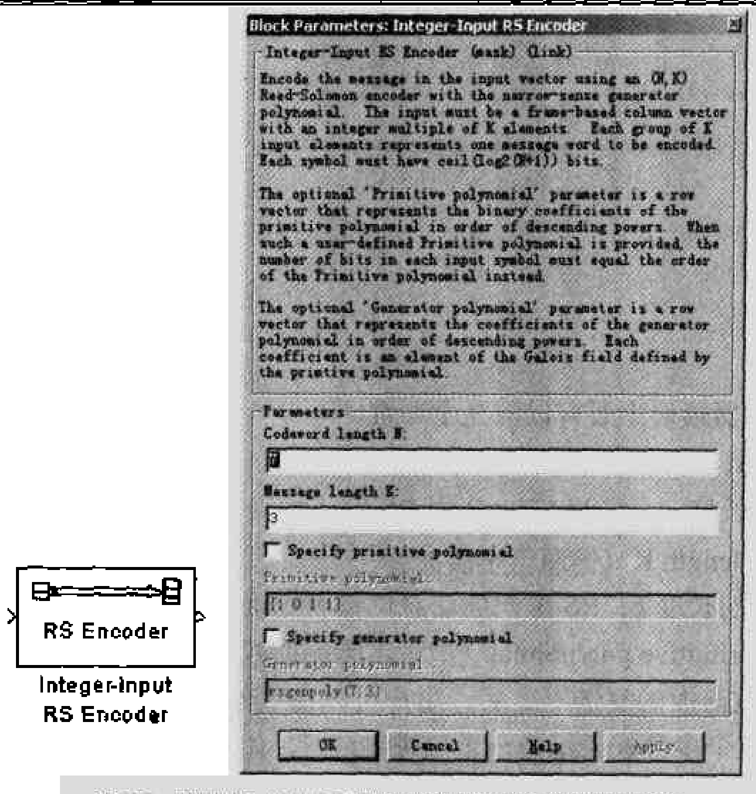



图 7-7　整型输入信号 RS 码编码器模块及其参数设置对话框

在 MATLAB 中，本原多项式可以通过函数 primpoly(x)来产生。对于 M 进制的 RS 编码，它的本原多项式的次数是$\log_2 M = \log_2(n+1)$。当码字长度 n 小于 M-1 时，RS 码的本原多项式的最高次数是$\lceil \log_2(N+1) \rceil$，因此，通过函数 primpoly(ceil($\log_2$(n+1)))可以产生码字长度为 n 的 RS 码的本原多项式。

RS 码的生成多项式可以用 MATLAB 函数[genpoly t]=rsgenpoly(n, k, poly)来产生，其中参数 n 表示 RS 码的码字长度，参数 k 表示信息位长度，poly 表示伽罗华域的本原多项式，两个输出参数 genpoly 和 t 分别表示生成多项式的系数和 RS 码的纠错能力。下面的程序用于产生一个（15，11）RS 码的本原多项式和生成多项式。

```
>> n=15;
>> primpoly(ceil(log2(n+1)),'min')
Primitive polynomial(s) =
 D^4+D^1+1
ans =
    19
>> primpoly(ceil(log2(n+1)),'all')
Primitive polynomial(s) =
D^4+D^1+1
D^4+D^3+1
ans =
    19
```

```
    25
>> [genpoly t]=rsgenpoly(15,11)
genpoly = GF(2^4) array. Primitive polynomial = D^4+D+1 (19 decimal)
Array elements =
    1    13    12    8    7
t =
    2
```

整型 RS 编码器模块主要有以下几个参数。

■ Codeword length N（码字长度）

RS 码的码字长度 $n$。在 MATLAB 中 RS 编码的码字长度应该大于 4 而小于 $2^{16}-1$。

■ Message length K（信息位长度）

RS 码信息位的长度 $k$。RS 编码的信息位长度应该小于码字的长度。

■ Specify primitive polynomial（指定本原多项式）

当选中该参数前面的复选框之后，用户可以在“Primitive polynomial”参数中指定相应的本原多项式，这个多项式表示为一个二进制向量。如果不指定本原多项式，MATLAB 将通过函数 primpoly($x$)产生一个最小的本原多项式。

■ Primitive polynomial（本原多项式）

当选中“Specify primitive polynomial”之后本参数有效，它是一个与本原多项式相对应的二进制向量。例如，本原多项式 $p(x)=x^3+x+1$ 表示成$[1\quad 0\quad 1\quad 1]$。

■ Specify generator polynomial（指定生成多项式）

当选中该参数前面的复选框之后，用户可以在“Generator polynomial”参数中指定相应的生成多项式，这个多项式表示为一个整型向量。如果不指定生成多项式，MATLAB 将使用 rsgenpoly()函数产生一个相应于最小本原多项式的生成多项式。

■ Generator polynomial（生成多项式）

当选中“Specify generator polynomial”之后本参数有效，它是一个与生成多项式相对应的整型行向量，其中的每个元素的取值范围是$[0,2^m-1]$。例如，对于（15，11）RS 码的生成多项式 $g(x)=x^4+(\alpha^3+\alpha^2+1)x^3+(\alpha^3+\alpha^2)x^2+\alpha^3x+(\alpha^2+\alpha+1)$ 表示成整型向量就是[1 13 12 8 7]。

### 2. 二进制 RS 码编码器

二进制 RS 编码器（Binary-Input RS Encoder）产生一个 $m$ 进制（$n$，$k$）Reed-Solomon 码。与整型 RS 编码器不同的是，二进制 RS 编码器的输入信号和输出信号都是帧格式的二进制信号序列，它用 $m$ 位二进制码来表示一个 $M$ 进制符号，其中 $M=2^m$。因此，输入信号的长度应该是 $m\times k$ 的倍数，同时输出信号的长度是 $m\times n$ 的倍数。图 7-8 所示是二进制 RS 码编码器模块及其参数设置对话框。

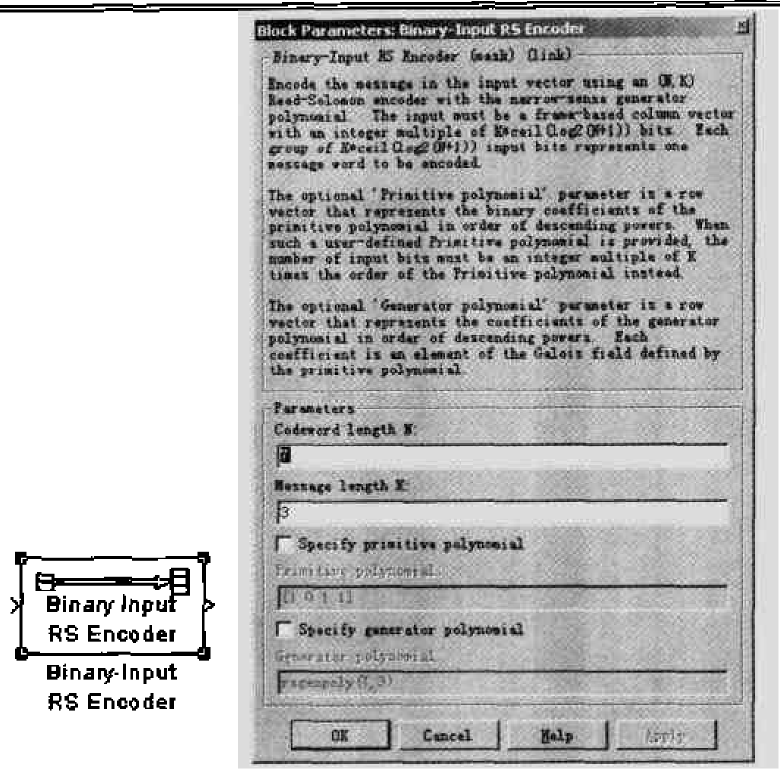



图 7-8　二进制输入信号 RS 码编码器模块及其参数设置对话框

二进制 RS 码编码器模块有以下几个参数。

■　Codeword length N（码字长度）

RS 码的码字长度 $n$。二进制 RS 码编码器的输出向量的长度等于 $m \times n$ 的倍数。

■　Message length K（信息位长度）

RS 码信息位的长度 $k$。二进制 RS 码编码器的输入向量的长度等于 $m \times k$ 的倍数。

■　Specify primitive polynomial（指定本原多项式）

当选中该参数前面的复选框之后，用户可以在“Primitive polynomial”参数中指定相应的本原多项式。

■　Primitive polynomial（本原多项式）

当选中“Specify primitive polynomial”之后本参数有效，它是一个与本原多项式相对应的二进制向量。

■　Specify generator polynomial（指定生成多项式）

当选中该参数前面的复选框之后，用户可以在“Generator polynomial”参数中指定相应的生成多项式。

■　Generator polynomial（生成多项式）

当选中“Specify generator polynomial”之后本参数有效，它是一个与生成多项式相对应的整型行向量，其中的每个元素的取值范围是 $[0, 2^m - 1]$。

### 3. 整型 RS 码解码器

整型 RS 解码器（Integer-Output RS Decoder）用来实现对 $M$ 进制（$n$，$k$）Reed-Solomon 码的解码，其中 $M = 2^m$。在解码过程中，整型 RS 解码器的各种参数设置应该与相应的整型

RS 编码器保持一致。整型 RS 解码器的输入信号和输出信号都是介于 0 和 $M$-1 之间的整数，输入信号的长度是 $n$ 的整数倍，输出信号的长度则是 $k$ 的整数倍。整型 RS 解码器模块及其参数设置对话框如图 7-9 所示。

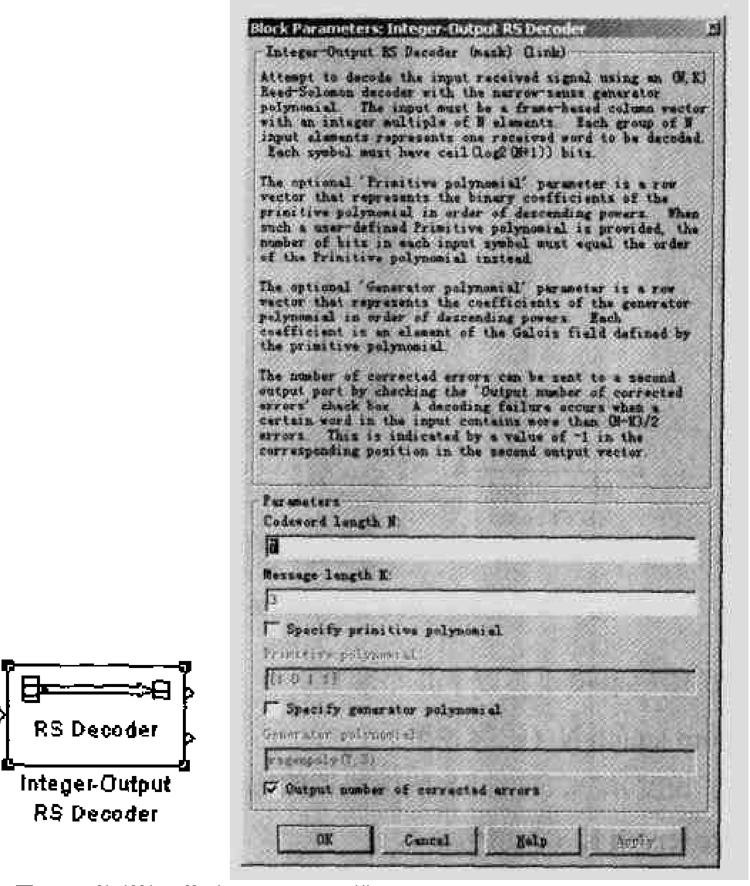



图 7-9　整型输入信号 RS 码解码器模块及其参数设置对话框

整型 RS 解码器可以有两个输出端口，其中第一个输出端口输出解码之后的信号，第二个输出端口则在选中 Output port for number of corrected errors 后有效，它表示解码过程中纠正的误码的个数。RS 码具有很强的纠错能力，一个（$n$，$k$）RS 码能够纠正 $\lfloor (n-k)/2 \rfloor$ 个 $M$ 进制符号的错误。当错误符号的个数超过了 RS 码的纠错能力时，第二输出端口的输出信号是 -1。整型 RS 解码器模块主要有以下几个参数。

■　Codeword length N（码字长度）

RS 码的码字长度 $n$。

■　Message length K（信息位长度）

RS 码信息位的长度 $k$。

■　Specify primitive polynomial（指定本原多项式）

当选中该参数前面的复选框之后，用户可以在“Primitive polynomial”参数中指定相应的本原多项式。

■　Primitive polynomial（本原多项式）

当选中“Specify primitive polynomial”之后本参数有效，它是一个与本原多项式相对应的二进制行向量。

■　Specify generator polynomial（指定生成多项式）

当选中该参数前面的复选框之后，用户可以在“Generator polynomial”参数中指定相应的生成多项式。

■　Generator polynomial（生成多项式）

当选中“Specify generator polynomial”之后本参数有效，它是一个与生成多项式相对应的整型行向量，其中的每个元素的取值范围是$\left[0,2^m-1\right]$。

■　Output port for number of corrected errors（输出误码的个数）

当选中了“Output port for number of corrected errors”复选框之后，整型 RS 解码器的第二输出端口输出当前码字的被纠正的误码的个数。如果不选择这个选项，整型 RS 解码器只有一个输出端口。

#### 4. 二进制 RS 码解码器

二进制 RS 解码器（Binary-Output RS Decoder）对以二进制序列型时表示的 $M$ 进制（$n$，$k$）Reed-Solomon 码的解码，其中 $M=2^m$。二进制 RS 解码器的输入信号和输出信号都是二进制符号，并且 $m$ 个二进制符号表示一个 $M$ 进制整数。二进制 RS 解码器只能处理帧格式数据，输入帧的长度是 $m\times n$ 的整数倍，输出帧的长度是 $m\times k$ 的整数倍。二进制 RS 解码器模块及其参数设置如图 7-10 所示。

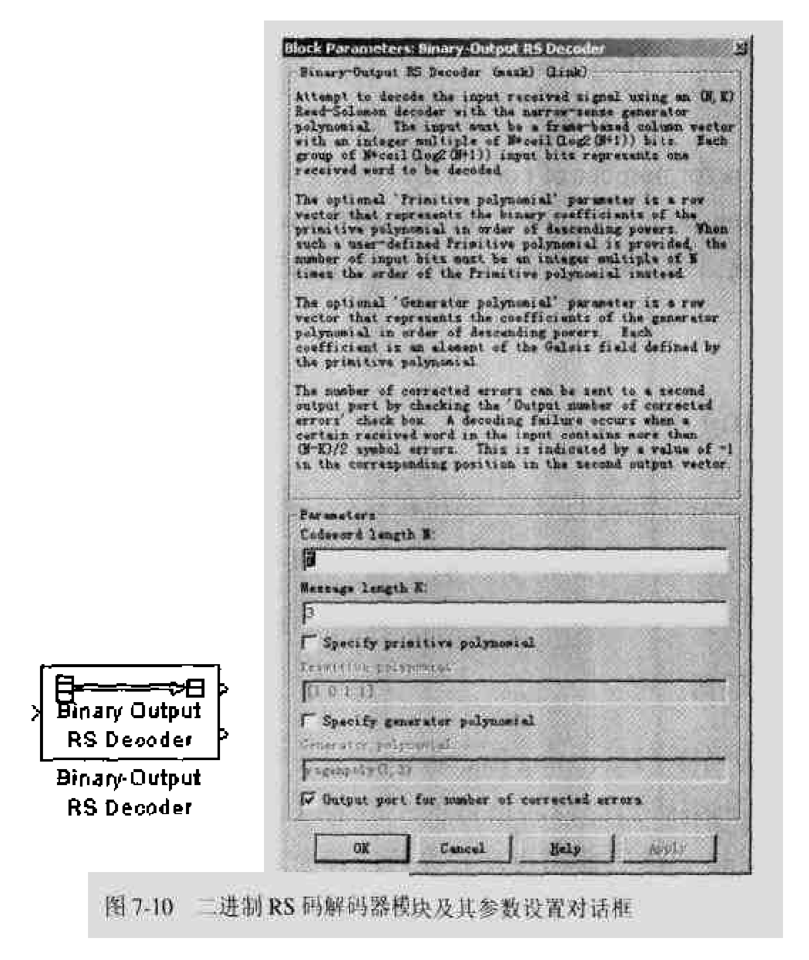



图 7-10　二进制 RS 码解码器模块及其参数设置对话框

选中 Output port for number of corrected errors 后，二进制 RS 解码器有两个输出端口，其

中第一个输出端口输出解码之后的信号，第二个输出端口是一个向量，它表示在解码过程中纠正的误码的个数。RS 码具有很强的纠错能力，一个（$n$，$k$）RS 码能够纠正$\lfloor (n-k)/2 \rfloor$个 $M$ 进制符号的错误。当错误符号的个数超过了 RS 码的纠错能力时，第二输出端口的输出信号是-1。二进制 RS 解码器主要有以下几个参数。

■ Codeword length N（码字长度）

RS 码的码字长度 $n$。二进制输入信号 RS 码解码器的输出向量的长度等于 $m \times n$。

■ Message length K（信息位长度）

RS 码信息位的长度 $k$。二进制输入信号 RS 码解码器的输入向量的长度等于 $m \times k$。

■ Specify primitive polynomial（指定本原多项式）

当选中该参数前面的复选框之后，用户可以在 Primitive polynomial 参数中指定相应的本原多项式。

■ Primitive polynomial（本原多项式）

当选中 Specify primitive polynomial 之后本参数有效，它是一个与本原多项式相对应的二进制向量。

■ Specify generator polynomial（指定生成多项式）

当选中该参数前面的复选框之后，用户可以在 Generator polynomial 参数中指定相应的生成多项式。

■ Generator polynomial（生成多项式）

当选中 Specify generator polynomial 之后本参数有效，它是一个与生成多项式相对应的整型行向量，其中的每个元素的取值范围是[0,$2^m-1$]。

■ Output port for number of corrected errors（输出误码的个数）

当选中了 Output port for number of corrected errors 前面的复选框之后，二进制输入信号 RS 解码器的第二输出端口输出当前码字中被纠正的误码的个数。如果不选择这个选项，二进制 RS 解码器只有一个输出端口。

## 7.1.5 Hamming 码

Hamming 码是一种能够纠正单个随机错误的线性码，其码字长度等于 $n=2^m-1$，信息位长度 $k=n-m=2^m-m-1$。对比 Hamming 码与 BCH 码可以发现，具有循环性质的 Hamming 码实质上就是能够纠正单个随机错误的本原 BCH 码。

### 1. Hamming 码编码器

Hamming 码编码器（Hamming Encoder）产生一个（$n$，$k$）Hamming 码，其中 $n=2^m-1$，$m \geq 3$。Hamming 码编码器的输入信号的长度为 $k$，输出信号的长度为 $n$，并且满足条件 $n=k+m$。Hamming 码编码器模块及其参数设置如图 7-11 所示。

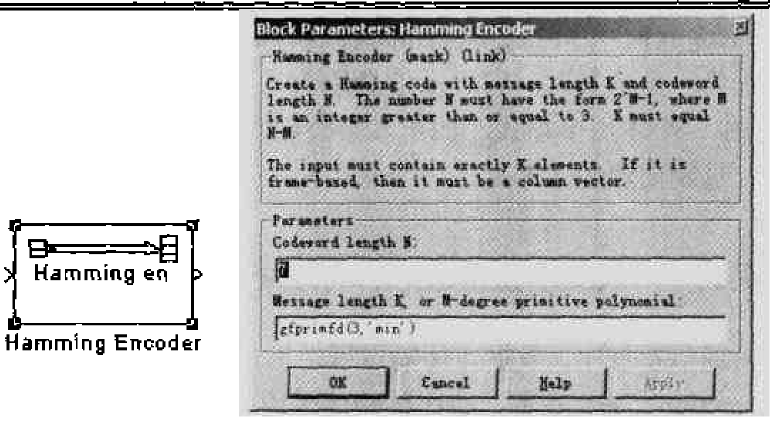



图 7-11 Hamming 码编码器模块及其参数设置对话框

Hamming 码编码器模块有两个参数。

- Codeword length N（码字长度）

Hamming 码的码字长度，它等于 Hamming 码编码器输出信号向量的长度。

- Message length K, or M-degree primitive polynomial（信息位长度/本原多项式）

当本参数设置为一个标量时，它表示信息位的长度，即输入信号向量的长度，这时候由 MATLAB 自动产生一个本原多项式。如果本参数设置为一个向量时，它是伽罗华域 $GF(2^m)$ 的本原多项式的二进制向量表示形式。通过 MATLAB 函数 gfprimfd(*m*, 'min')可以找到一个最小的本原多项式。

### 2. Hamming 码解码器

Hamming 码解码器（Hamming Decoder）用于对（*n*，*k*）Hamming 码进行解码，得到原始的信息位序列。Hamming 码解码器的参数设置应该与相应的 Hamming 码编码器参数保持一致。Hamming 码解码器的输入信号的长度为 $n$（$n=2^m-1$，$m\geq3$），输出信号的长度为 $k$，且 $n=k+m$。Hamming 码解码器模块及其参数设置对话框如图 7-12 所示。

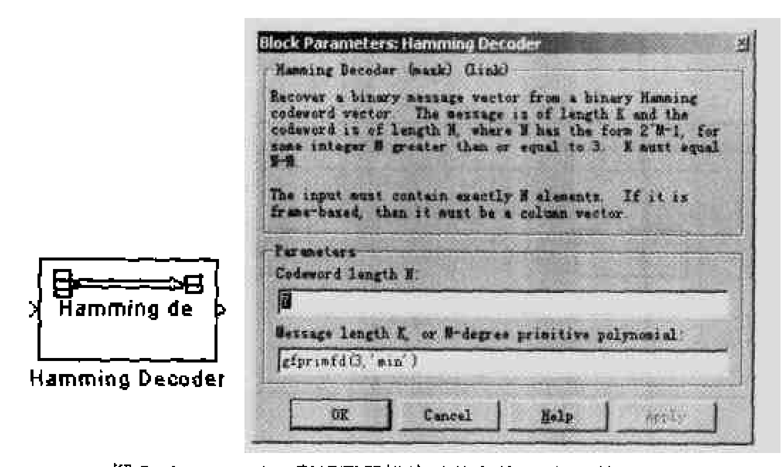



图 7-12 Hamming 码解码器模块及其参数设置对话框

Hamming 码解码器主要由以下几个参数。

- Codeword length N（码字长度）

Hamming 码的码字长度，它等于 Hamming 码解码器输入信号向量的长度。

■ Message length K, or M-degree primitive polynomial（信息位长度/本原多项式）

当本参数设置为一个标量时，它表示信息位的长度，即输出信号向量的长度，这时候由 MATLAB 自动产生一个本原多项式。如果本参数设置为一个向量时，它是伽罗华域 $GF(2^m)$ 的本原多项式的二进制向量表示形式。

## 7.1.6 实例 7.1——Reed-Solomon 码在 CT2 中的应用

在早期的移动通信系统中，数据在传输过程中出现错误之后一般是通过前向纠错（FEC，Forward Error Correction）来纠正的，即在传输信号中引入一定的冗余度来实现信号的纠错功能。由于 Reed-Solomon 编码具有较强的纠错能力，因而在移动通信系统中得到了广泛的应用。需要强调的是，我们不能指望一种编码方法能够解决通信系统的传输质量问题。当接收信号的误比特数超过了编码的纠错能力和检错能力时，还需要通过其他方法来实现对信号的正确接收。在本实例中，我们将结合第二代无绳电话系统（CT2，2nd generation Cordless Telephone）中数据业务采用的 Reed-Solomon 编码来介绍 RS 编码的应用。

CT2 最初起源于英国，主要应用于家庭、商业以及公共电话系统中，一般使用 864.1 MHz 和 868.1 MHz 之间的频段（我国为 839～843MHz），采用 FDMA 方式，每个信道占用 100 kHz 的带宽，可支持 40 个信道。CT2 系统中基站的发射功率较小，手机最大发送功率为 10 dBw，调制方式为频移键控（FSK），标准覆盖半径为 40～200 米。

与第一代无绳电话系统相似地，CT2 一般用于进行语音通信，但是由于它采用了数字通信技术，因而也可以用来实现数据通信。CT2 的传输速率最高能达到 32kbit/s，这在一定程度上能够满足系统内的控制数据以及各种低速数据业务（如传真业务等）的要求。

CT2 采用时分双工（TDD，Time Division Duplex）方式，前向信道和反向信道以时分方式公用一个物理信道。CT2 中每个时隙的间隔是 2 毫秒，前向信道和反向信道各占用 1 毫秒。一般情况下，CT2 数据帧的长度等于 64 个字节（其中包括 44 字节的信息位、19 字节的校验位以及 1 个字节的帧同步标识 01111110），它分别在 8 个连续的时隙中传输，如图 7-13 所示。

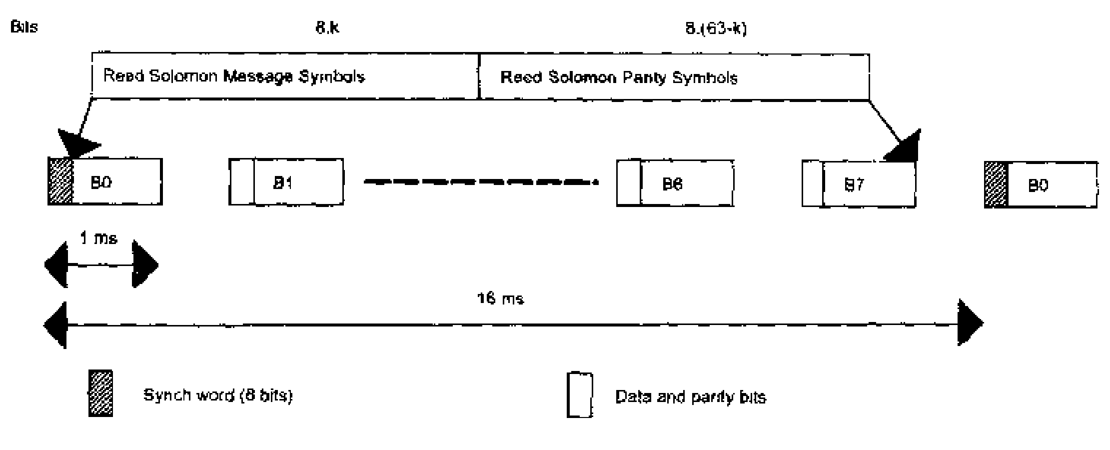



图 7-13 CT2 数据帧的传输方式

CT2 数据帧采用 RS 编码，信息位长度等于 44 字节，其中包括 4 个字节的控制信息开销，如图 7-14 所示。信息位经过 RS 编码之后产生 60 字节的编码信号，除此之外还有 3 个字节

用于检错，加上一个字节的帧同步标志，每帧长度是 64 个字节。

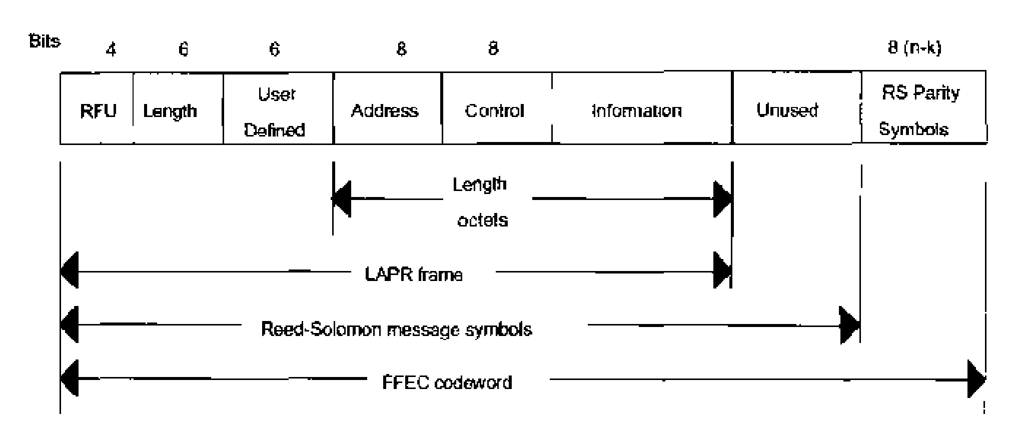



图 7-14 CT2 数据业务的帧结构

实例 7.1 对 CT2 数据业务中采用的 RS 编码进行仿真，整个系统由 3 个部分组成：Source（信源模块）、Binary Symmetric Channel（信道模块）以及 Sink（信宿模块）。信源模块产生一个随机的二进制数据帧，这些数据帧通过 RS 编码之后进入二进制对称信道，然后在信宿模块中进行 RS 解码，最后计算编码信号的误比特率，绘制信号误比特率与信道质量之间的关系图。实例 7.1 的系统结构图如图 7-15 所示。

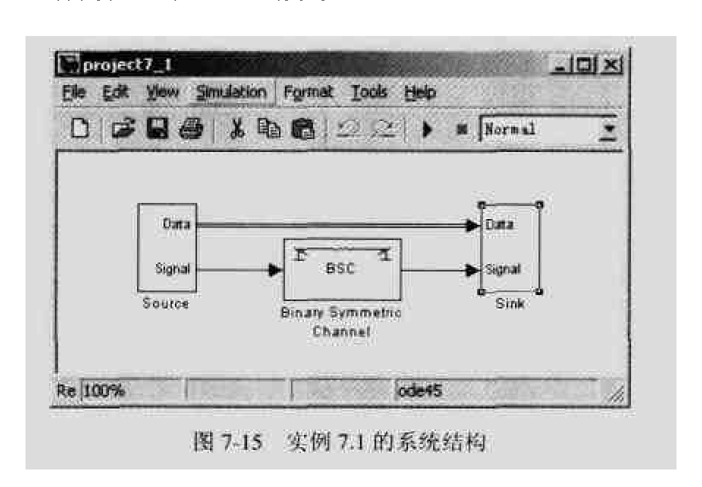



图 7-15 实例 7.1 的系统结构

图 7-16 所示是信源模块的系统结构。在信源模块中，Random Integer Generator（随机整数产生器）每隔 2 毫秒产生一个长度为 44 个字节的数据帧，这个数据帧经过 Integer-Input RS Encoder（RS 编码器）进行编码，编码后每个数据帧的长度等于 60 个字节。CT2 每帧数据的长度等于 64 个字节，其中有 4 个字节的开销，在这里我们用一个填充模块（Pad）把每帧数据的长度补齐为 64 个字节。最后，为了使编码信号能够通过二进制对称信道，还应该把每帧数据中的整数转换成二进制序列。Frame Status Conversion（帧状态转换模块）把帧格式的列向量转换成抽样格式的列向量，再由 MATLAB 函数模块（MATLAB Fcn）中的 de2bi($u$,8)把每个抽样只转换成 8 位二进制数。

信源模块有两个输出端口，其中 Data（第一个输出端口）输出编码前的帧数据，用于对误比特率进行统计：Signal（第二个输出端口）输出编码后的信号，它产生信道模块的输入信号。信源模块中的随机整数产生器、RS 编码器、填充模块、帧状态转换模块以及 MATLAB 函数模块的参数设置分别如表 7-1~表 7-5 所示。

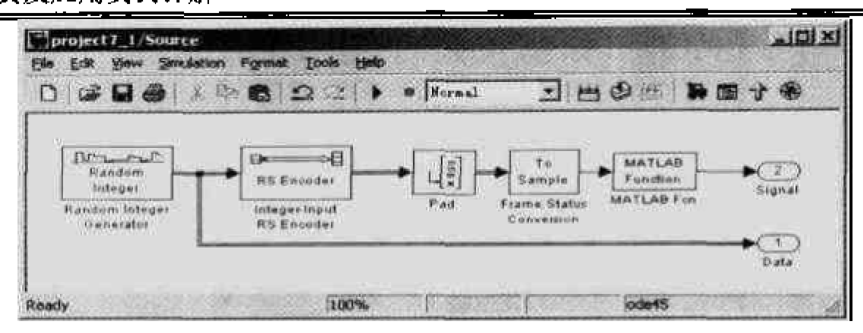



图 7-16　信源模块的系统结构

表 7-1　**Random Integer Generator（随机整数产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Random Integer Generator |
| M-ary number | 256 |
| Initial seed | 37 |
| Sample time | 1/BitRate |
| Frame-based outputs | Checked |
| Samples per frame | MessageLength |

表 7-2　**Integer-Input RS Encoder（RS 编码器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Integer-Input RS Encoder |
| Codeword length N | CodewordLength |
| Message length K | MessageLength |
| Specify primitive polynomial | Checked |
| Primitive Polynomial | PrimitivePolynomial |
| Specify generator polynomial | Unchecked |

表 7-3　**Pad（填充模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Pad |
| Value | 126 |
| Pad signal at | End |
| Pad along | Columns |
| Number of output rows | User-specified |
| Specified number of output rows | FrameLength |
| Action when truncation occurs | None |

表 7-4　**Frame Status Conversion（帧状态转换模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Frame Status Conversion |
| Inherit output frame status from Ref input port | Unchecked |
| Output signal | Sample-based |

表 7-5　**MATLAB Fcn（MATLAB 函数模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | MATLAB Fcn |
| MATLAB function | de2bi(u, 8) |
| Output dimensions | -1 |
| Output signal type | auto |
| Collapse 2-D results to 1-D | Unchecked |

Binary Symmetric Channel（二进制对称信道模块）在信源模块输出的编码信号中引入二进制随机错误，它以一定的概率改变某些二进制信号的数值，这个概率由二进制对称信道模块的参数 ChannelErrorRate 确定。二进制对称信道模块的参数设置如表 7-6 所示。

表 7-6　**Binary Symmetric Channel（二进制对称信道模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Binary Symmetric Channel |
| Error probability | ChannelErrorRate |
| Initial seed | 71 |
| Output error vector | Unchecked |

信宿模块如图 7-17 所示。信宿模块接收到的信号是一个 64 行 8 列的矩阵，MATLAB 函数模块通过调用 de2bi(*u*,8)把其中的每一个行向量转换成一个整数，然后通过 Frame Status Conversion（帧状态转换模块）和 Pad（填充模块）得到一个长度为 60 的帧结构的列向量。Integer-Output RS Decoder（RS 解码器模块）对这个数据帧进行译码，产生的输出信号是长度为 44 的数据帧。

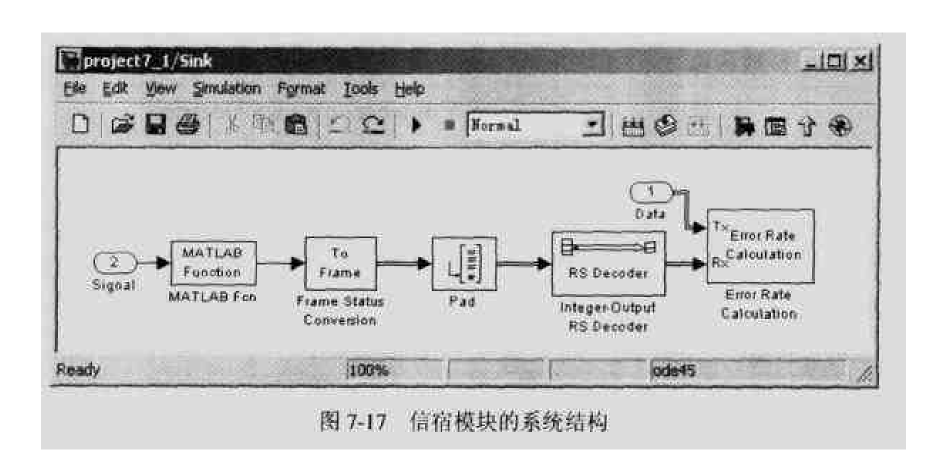


图 7-17　信宿模块的系统结构

信宿模块中的 Error Rate Calculation（误比特率统计模块）把 RS 解码信号和编码之前的信号进行比较，统计得到误比特率、误比特总数以及输入信号数，并且把这些信息保存在工作区变量 SymbolErrorRate 中。信宿模块的 MATLAB 函数调用模块、帧状态转换模块、填充模块、RS 解码器模块以及误比特率统计模块的参数设置分别如表 7-7~表 7-11 所示。

表 7-7　　MATLAB Fcn（MATLAB 函数模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | MATLAB Fcn |
| MATLAB function | bi2de(u) |
| Output dimensions | -1 |
| Output signal type | real |
| Collapse 2-D results to 1-D | Checked |

表 7-8　　Frame Status Conversion（帧状态转换模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Frame Status Conversion |
| Inherit output frame status from Ref input port | Unchecked |
| Output signal | Frame-based |

表 7-9　　Pad（填充模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Pad |
| Value | 1 |
| Pad signal at | End |
| Pad along | Columns |
| Number of output rows | User-specified |
| Specified number of output rows | CodewordLength |
| Action when truncation occurs | None |

表 7-10　　Integer-Input RS Decoder（RS 解码器）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Integer-Input RS Decoder |
| Codeword length N | CodewordLength |
| Message length K | MessageLength |
| Specify primitive polynomial | Checked |
| Primitive Polynomial | PrimitivePolynomial |
| Specify generator polynomial | Unchecked |
| Output number of corrected errors | Unchecked |

表 7-11　　Error Rate Calculation（误比特率统计模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Error Rate Calculation |
| Receive delay | 0 |
| Computation delay | 0 |
| Computation mode | Entire frame |

续表

| 参数名称 | 参数值 |
| --- | --- |
| Output data | Workspace |
| Variable name | SymbolErrorRate |
| Reset port | Unchecked |
| Stop simulation | Unchecked |

最后我们编写一个 M 文件，用于对模块中使用的各个变量的数值进行定义，同时循环执行仿真程序 project7_1，根据产生的一组数据绘制信道误比特率与编码信号误比特率之间的关系曲线图。M 文件 project7_1main.m 的程序代码如下：

```
% x 表示信道的误比特率
x=[0.0001 0.001 0.005 0.01 0.02 0.03 0.04 0.05 0.1 0.2];
% y 表示编码信号的误比特率，它的长度与 x 相同
y=x;

% 信源产生信号的 bit 率等于 22kbit/s
BitRate=22000;
% 仿真时间设置为 1 秒
SimulationTime=1;
% RS 编码器的码字长度（字节数）
CodewordLength = 60;
% RS 编码器输入信号的长度（字节数）
MessageLength = 44;
% RS 编码器的本原多项式
PrimitivePolynomial = [1 0 0 0 1 1 1 0 1];
% CT2 每个数据帧的长度（字节数）
FrameLength = 64;

% 循环执行仿真程序
for i=1:length(x)
    % 信道的误比特率依次取 x 中的元素
    ChannelErrorRate=x(i);
    % 运行仿真程序，得到的误比特率保存在工作区变量 SymbolErrorRate 中
    sim('project7_1');
    % 计算 SymbolErrorRate 的均值作为本次仿真的误比特率
    y(i)=SymbolErrorRate(1);
end
% 绘制 x 和 y 的关系曲线图，纵坐标采用对数坐标
loglog(x,y);
```

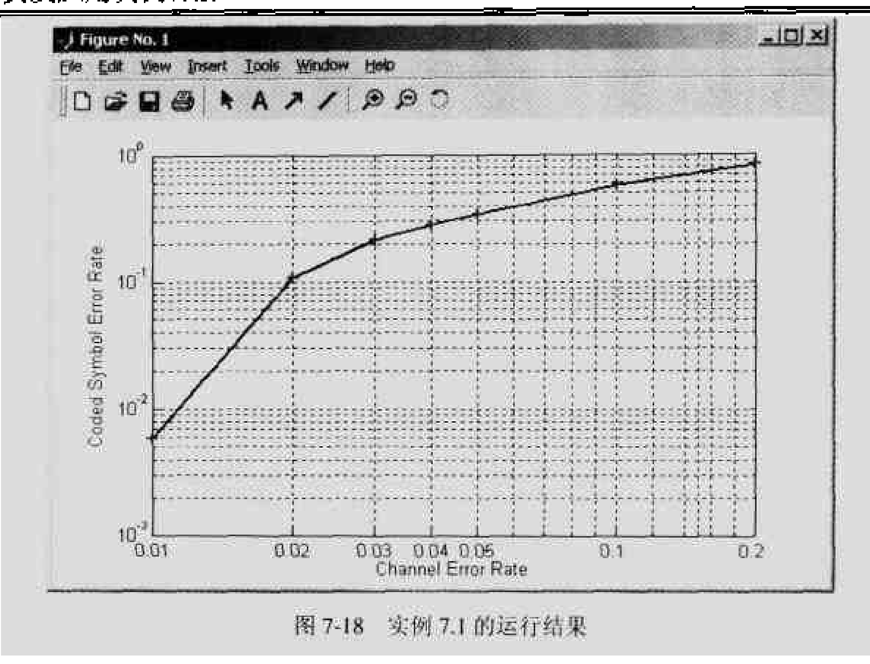



图 7-18 实例 7.1 的运行结果

图 7-18 所示是运行 M 文件 project7_1main 后得到的仿真结果。我们在 M 文件中设置了 3 个小于 1%的信道误比特率作为测试点，但是在我们这个有限的仿真时间中，相应于这些信道误比特率的编码信号误比特率都等于 0，因此这些点没有能够在图中表现出来。从图中可以看到，只有当信道的误比特率低于 1%时，RS（60，44）编码才能取得较好的效果。信道误比特率增大之后，接收信号中的误码个数将超过 RS 编码的纠错能力，因此，当信道误比特率较高时，还需要联合采用其他方法才能够保证信号的正确接收。

## 7.2 循环冗余码

循环冗余码 CRC（Cyclic redundancy check）是一种使用相当频繁的检错码。与分组码和卷积码不同的是，循环冗余码不具有纠错能力。当接收端检测到传输错误时，它并不去纠正这个传输错误，而是要求发送端重新发送这个信号序列。在循环冗余码的编码过程中，发送段对每一个特定长度的信息序列计算得到一个循环冗余码，并且把这个循环冗余码附加到原始的信息序列的末尾一起发送出去。接收端接收到带有循环冗余码的信号后，从中分离出信息位序列和循环冗余码，然后根据接收到的信息位序列重新计算循环冗余码。如果这个重新计算得到的循环冗余码与分离出来的循环冗余码不同，则接收信号序列存在着传输错误。这时候接收端会要求发送端的重新发送这个信号序列，通过这个过程实现对信号的纠错。

### 7.2.1 CRC 编码器

在 MATLAB 中，CRC 编码器有两种，即通用 CRC 编码器和 CRC-N 编码器，这两个 CRC 编码器比较接近，它们之间的区别在于，后者提供了 6 个常用的 CRC 生成多项式，使用起来比较方便。CRC 编码的一般过程如下：把输入的数据左移 r 位，然后除以生成多项式 P，得到一个余式，这个余式就是 CRC 编码产生的循环冗余码。把这个余式附加到原始的信息序列的末尾，就得到了经过 CRC 编码的输出信号序列。

## 1．通用 CRC 编码器

通用 CRC 编码器（General CRC Generator）根据输入的一帧数据计算得到这帧数据的循环冗余码 CRC（cyclic redundancy code），并且把这个循环冗余码附加到帧数据的后面，形成输出数据流。通用 CRC 编码器有一个输入端口和一个输出端口，并且它的输入数据和输出数据都应该是帧格式的二进制列向量。通用CRC编码器模块及其参数设置对话框如图7-19所示。

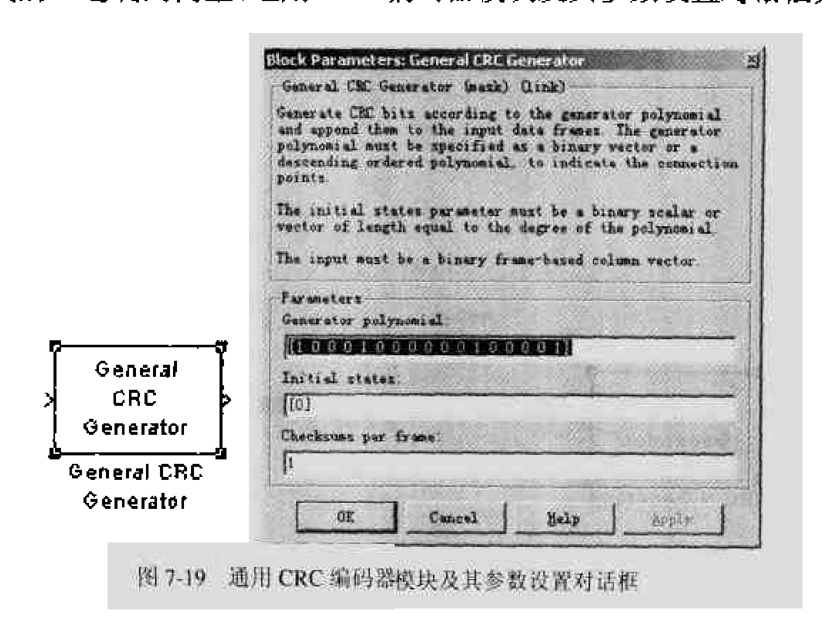



图 7-19　通用 CRC 编码器模块及其参数设置对话框

如果通用 CRC 编码器的输入数据的帧长度等于 $n$，生成多项式的最高次数为 $r$，对每帧输入数据产生 $k$ 个校验和（checksum），则通用 CRC 编码器的工作流程如下：

（1）把输入的一帧数据等分成 $k$ 个部分，每个部分 $w_i$ 的长度是 $n/k$；

（2）在每个部分的数据 $w_i$ 后面添加 $r$ 个二进制位，并且这 $r$ 个二进制位的数值等于通用 CRC 编码器的初始状态，得到二进制序列 $s_i$；

（3）计算 $s_i$ 的循环冗余码 $c_i$；

（4）把循环冗余码 $c_i$ 添加到 $w_i$ 后面，得到二进制序列 $u_i$；

（5）把所有的 $u_i$ 连接起来形成输出数据帧。

通用 CRC 编码器模块有如下 3 个参数。

■　Generator polynomial（生成多项式）

通用 CRC 编码器的生成多项式。这个生成多项式有两种表示方式：用生成多项式的系数组成的二进制向量，或者是由生成多项式中系数不等于 0 的项的指数组成的整型向量。例如，对于生成多项式 $x^3+x^2+1$，既可以表示为二进制向量[1 1 0 1]，也可以表示成整型向量[3 2 0]。

■　Initial states（初始状态）

本参数用于确定通用 CRC 编码器中移位寄存器的初始状态。当本参数是一个向量时，它的长度等于通用 CRC 编码器的生成多项式的最高次数；当本参数是一个标量时，MATLAB 自动把这个标量扩展成一个向量，向量的长度等于通用 CRC 编码器的生成多项式的最高次数，并且向量中的每个元素都等于这个标量。

■　Checksums per frame（每帧的校验和的个数）

本参数用于指定每帧数据产生的校验和的个数。假设每帧的校验和的个数等于 $k$，则每帧输入数据的长度应该是 $k$ 的整数倍。

## 2. CRC-N 编码器

CRC-N Generator（CRC-N 编码器）计算每一个输入信号帧的循环冗余码（CRC），并且把计算得到的循环冗余码附加到输入帧的末尾。CRC-N 编码器是通用 CRC 编码器的简化，它的工作方式与通用 CRC 编码器类似，但是它提供了若干个经常使用的生成多项式，N 就表示这些生成多项式的最高次数。CRC-N 编码器模块及其参数设置对话框如图 7-20 所示。

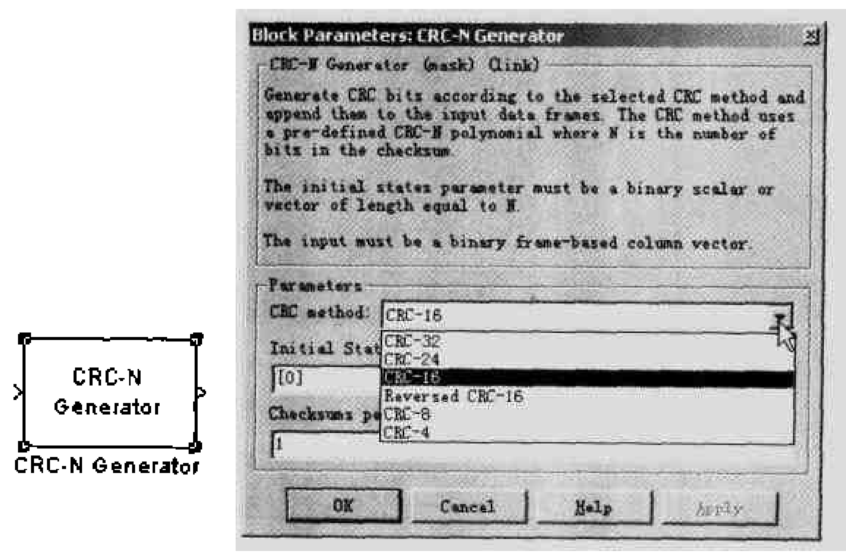



图 7-20　CRC-N 编码器模块及其参数设置对话框

■　CRC-N method（CRC-N 模式）

本参数用于确定 CRC-N 编码器使用的生成多项式。可供选择的生成多项式有 6 种，如表 7-12 所示。

表 7-12　CRC-N 编码器的生成多项式

| CRC-N 模式 | 生成多项式 | 位数 |
|---|---|---|
| CRC-32 | $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$ | 32 |
| CRC-24 | $x^{24}+x^{23}+x^{14}+x^{12}+x^{8}+1$ | 24 |
| CRC-16 | $x^{16}+x^{15}+x^{2}+1$ | 16 |
| Reversed CRC-16 | $x^{16}+x^{14}+x+1$ | 16 |
| CRC-8 | $x^{8}+x^{7}+x^{6}+x^{4}+x^{2}+1$ | 8 |
| CRC-4 | $x^{4}+x^{3}+x^{2}+x+1$ | 4 |

■　Initial states（初始状态）

本参数用于确定 CRC-N 编码器中移位寄存器的初始状态。当本参数是一个向量时，它的长度等于 CRC-N 编码器的生成多项式的最高次数；当本参数是一个标量时，MATLAB 自动把这个标量扩展成一个向量，向量的长度等于 CRC-N 编码器的生成多项式的最高次数，并且向量中的每个元素都等于这个标量。

■　Checksums per frame（每帧的校验和的个数）

本参数用于指定每帧数据产生的校验和的个数。假设每帧的校验和的个数等于 $k$，则每帧输入数据的长度应该是 $k$ 的整数倍。

## 7.2.2　CRC 检测器

与通用 CRC 生成器和 CRC-N 生成器相对应地，CRC 检测器也有两种：通用 CRC 检测器和 CRC-N 检测器。这两种检测器具有相同的工作原理，它们首先从接收到的二进制序列中分离出信息序列和 CRC，然后根据接收端的信息序列重新计算 CRC。如果重新计算得到的 CRC 与接收到的 CRC 相等，则认为接收序列是正确的；否则，接受序列存在着传输错误。

### 1．通用 CRC 检测器

通用 CRC 检测器（General CRC Syndrome Detector）从接收到的信号中分离出信息序列和 CRC，然后对信息序列重新计算 CRC。如果重新计算得到的 CRC 与分离出来的 CRC 相等，则认为接收到的信号是正确的；否则，接收到的信号存在着传输错误。通用 CRC 检测器模块及其参数设置对话框如图 7-21 所示。

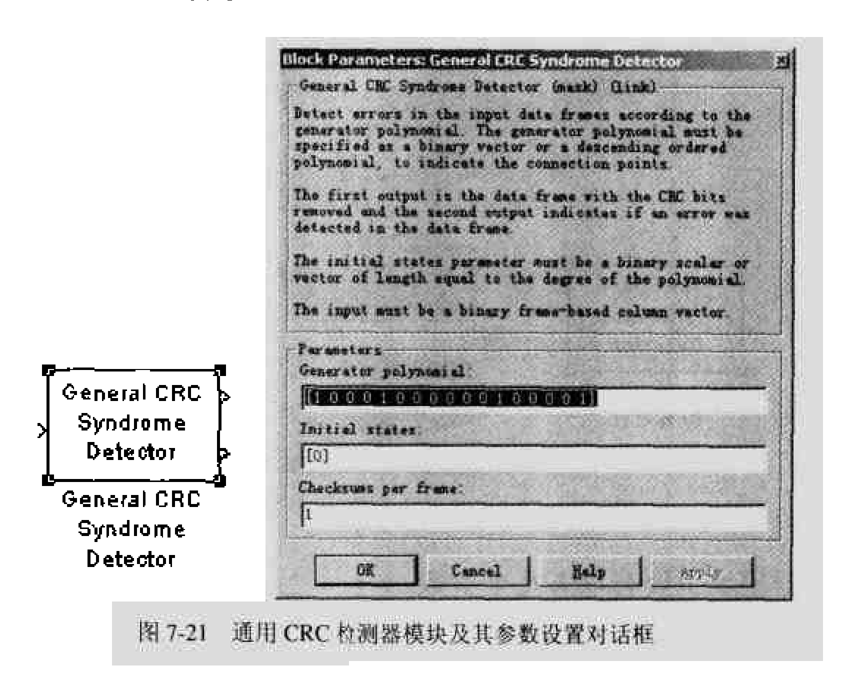



图 7-21　通用 CRC 检测器模块及其参数设置对话框

通用 CRC 检测器模块有两个输出端口：第一个输出端口的输出信号是除去了 CRC 的信息序列；第二个输出端口的输出信号表示对接收信号的 CRC 进行校验的结果。第二输出端口的输出信号是一个向量。如果根据信息位重新计算得到的 CRC 与接收到的 CRC 相等，则输出信号等于 0；否则，输出信号等于 1。

通用 CRC 检测器主要有以下几个参数。

■　Generator polynomial（生成多项式）

与通用 CRC 检测器相对应的 CRC 编码器的生成多项式。这个生成多项式有两种表示方式：用生成多项式的系数组成的二进制向量，或者是由生成多项式中系数不等于 0 的项的指数组成的整型向量。

■　Initial states（初始状态）

本参数用于确定与通用 CRC 检测器相对应的 CRC 编码器中移位寄存器的初始状态。当

本参数是一个向量时，它的长度等于 CRC 编码器的生成多项式的最高次数；当本参数是一个标量时，MATLAB 自动把这个标量扩展成一个向量，向量的长度等于 CRC 编码器的生成多项式的最高次数，并且向量中的每个元素都等于这个标量。

- Checksums per frame（每帧的校验和的个数）

本参数用于指定每帧数据产生的校验和的个数。假设每帧的校验和的个数等于 $k$，则每帧输入数据的长度应该是 $k$ 的整数倍。

### 2. CRC-N 检测器

CRC-N 检测器（CRC-N Syndrome Detector）从接收到的信号中分离出信息序列和 CRC，然后对信息序列重新计算 CRC。如果重新计算得到的 CRC 与分离出来的 CRC 相等，则认为接收到的信号是正确的；否则，接收到的信号存在着传输错误。CRC-N 检测器模块及其参数设置对话框如图 7-22 所示。

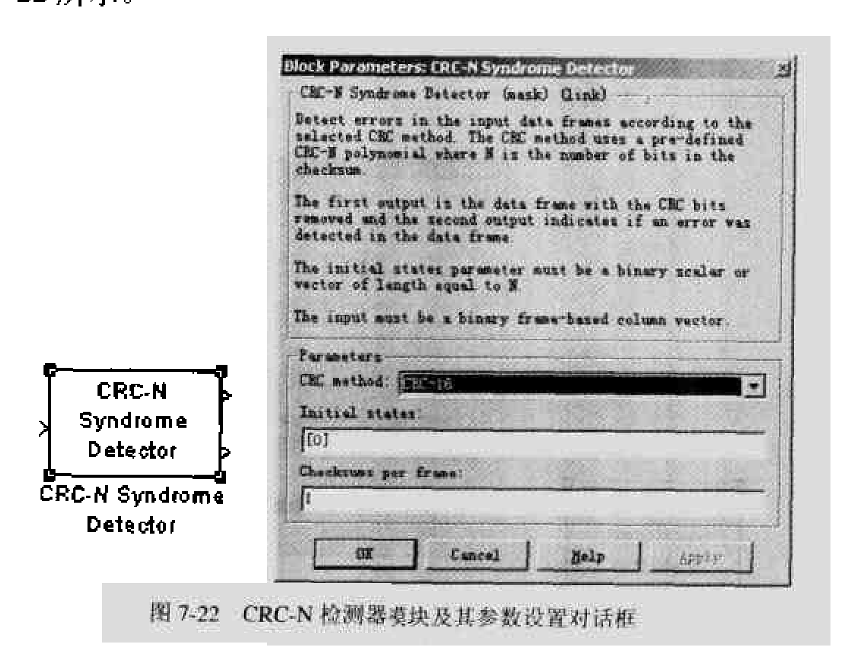



图 7-22　CRC-N 检测器模块及其参数设置对话框

CRC-N 检测器模块主要有以下几个参数。

- CRC-N method（CRC-N 模式）

本参数用于确定与 CRC-N 检测器相对应的 CRC-N 编码器使用的生成多项式。可供选择的生成多项式如表 7-12 所示。

- Initial states（初始状态）

本参数用于确定与 CRC-N 检测器相对应的 CRC-N 编码器中移位寄存器的初始状态。当本参数是一个向量时，它的长度等于 CRC-N 编码器的生成多项式的最高次数；当本参数是一个标量时，MATLAB 自动把这个标量扩展成一个向量，向量的长度等于 CRC-N 编码器的生成多项式的最高次数，并且向量中的每个元素都等于这个标量。

- Checksums per frame（每帧的校验和的个数）

本参数用于指定每帧数据产生的校验和的个数。假设每帧的校验和的个数等于 $k$，则每帧输入数据的长度应该是 $k$ 的整数倍。

### 7.2.3 实例 7.2——CRC-16 编码在 DECT 中的应用及其性能

DECT（Digital European Cordless Telephone）是一种新的无绳电话通信标准，它是由欧洲电信标准化协会（ETSI，European Telecommunications Standards Institute）于 1992 年在 ISDN 和 GSM 的基础上制订的。DECT 可用于家庭、商业通信系统及公共网的接入。DECT 除用于话音通信外，还可传输数据，速率达 552Kbit/s。DECT 使用的频段介于 1880 MHz 和 1900MHz 之间，该频段划分为 10 个载波，每个载波具有 12×2 个时隙，共有 120 个双工信道，频道间隔为 1728 KHz。DECT 手机最大发射功率为 24dBw，标准覆盖半径为 30 到 200 米。

DECT 的接入方式为 TDMA/TDD，每个物理信道分成 24 个时隙，这 24 个时隙称为一帧，帧长为 10 毫秒。DECT 帧中的 24 个时隙可供 6 个用户同时使用，每个用户占用两个时隙（分别用于前向信道和反向信道）。因此，在 DECT 中，用户发送和接收数据帧的间隔是 10 毫秒。每个时隙可传输 480 个比特信息，其中包括 32bit 的同步前缀，388bit 的数据以及 60bit 的保护间隔。在这 388bit 的数据中，DECT 只对 48bit 的头部进行 CRC 校验，产生 16bit 的校验位，剩余的 320bit 用于直接传输数据。本节将结合 DECT 介绍 CRC 编码的应用及其性能。

实例的系统结构如图 7-23 所示。信源模块（Source）产生一个每隔 10 毫秒产生一个数据帧，这个数据帧通过 CRC 校验和填充之后长度等于 480bit。数据帧在 Channel（信道模块）中传输，这个信道是一个二进制对称信道。Sink（信宿模块）对接收到的信号进行 CRC 译码。在这里我们关心的是 CRC 的校验能力，因此我们对 CRC 编码之前的信号和编码之后的信号进行比较，检查这个数据帧中是否存在着传输错误。同时，CRC 检测器也产生一个关于这个数据帧的质量报告。如果这两种判断方法得到的结果相同，则表示 CRC 的检测结果是正确的；否则，要么是传输过程中发生的错误比特的个数超过了 CRC 的检测能力，要么是信息位本身没有错误，但是 CRC 校验位出现了传输错误导致错误的判决。这样，我们就能够获得错误判决帧所占的比例与信道误比特率之间的关系。

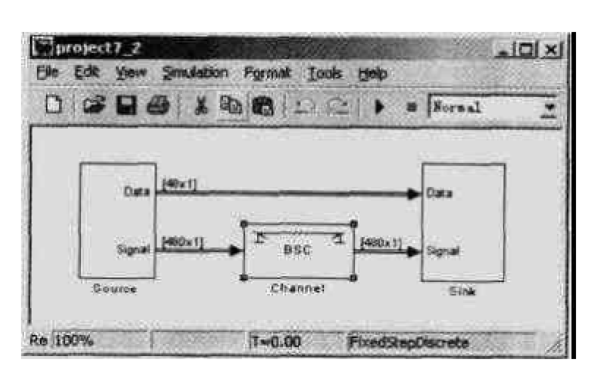



图 7-23 实例 7.2 的系统结构

图 7-24 所示是信源模块的结构框图。信源模块中的 Bernoulli Binary Generator（贝努利二进制产生器）每隔 10 毫秒产生一个长度为 48bit 的数据帧，这个数据帧相当于 DECT 数据帧中的受保护的信息头。这个 48bit 的数据帧通过 CRC-N Generator（CRC 编码器模块）后，产生 16bit 的 CRC 校验位，这时候每个数据帧的长度变成 64bit。由于 DECT 的数据帧的长度是 480bit，因此编码后的数据帧还需要通过一个 Zero Pad（零填充模块）。

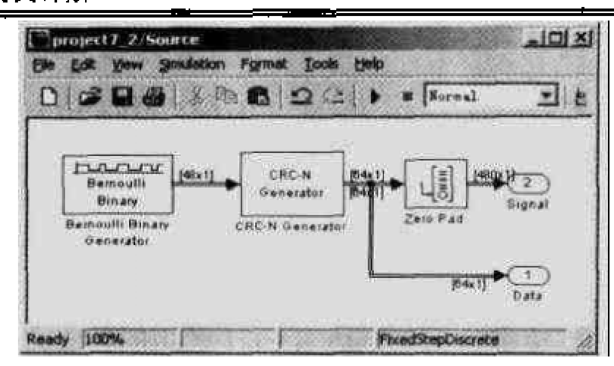



图 7-24 信源模块的系统结构

信源模块输出 CRC 编码前的信号（Data）和编码后的信号（Signal），其中编码前的信号直接进入信宿模块，用于计算误比特率，编码后的信号则在信道模块中传输。贝努利二进制产生器、CRC 编码器模块以及零填充模块的参数设置分别如表 7-13～7-15 所示。

**表 7-13 Bernoulli Binary Generator（贝努利二进制产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | 61 |
| Sample time | BitPeriod |
| Frame-based outputs | Checked |
| Samples per frame | ProtectedData |

**表 7-14 CRC-N Generator（CRC 编码器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | CRC-N Generator |
| CRC method | CRC-16 |
| Initial States | [0] |
| Checksums per frame | 1 |

**表 7-15 Zero Pad（零填充模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Zero Pad |
| Pad signal at | End |
| Pad along | Columns |
| Number of output rows | User-specified |
| Specified number of output rows | FrameLength |
| Action when truncation occurs | None |

Binary Symmetric Channel（二进制对称信道模块）在包含了 CRC 编码的数据帧中引入二进制随机错误，它以一定的概率改变某些二进制信号的数值，这个概率由二进制对称信道模块的参数 ChannelErrorRate 确定。二进制对称信道模块的参数设置如表 7-16 所示。

表 7-16　　**Binary Symmetric Channel（二进制对称信道模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Binary Symmetric Channel |
| Error probability | ChannelErrorRate |
| Initial seed | 71 |
| Output error vector | Unchecked |

信宿模块比较复杂，它需要对 CRC 编码信号进行检测，然后统计误比特率，计算出 CRC 未能检测到的传输错误所占的比例，然后绘制这个比例与信道的误比特率之间的关系曲线图。信宿模块的系统结构如图 7-25 所示。

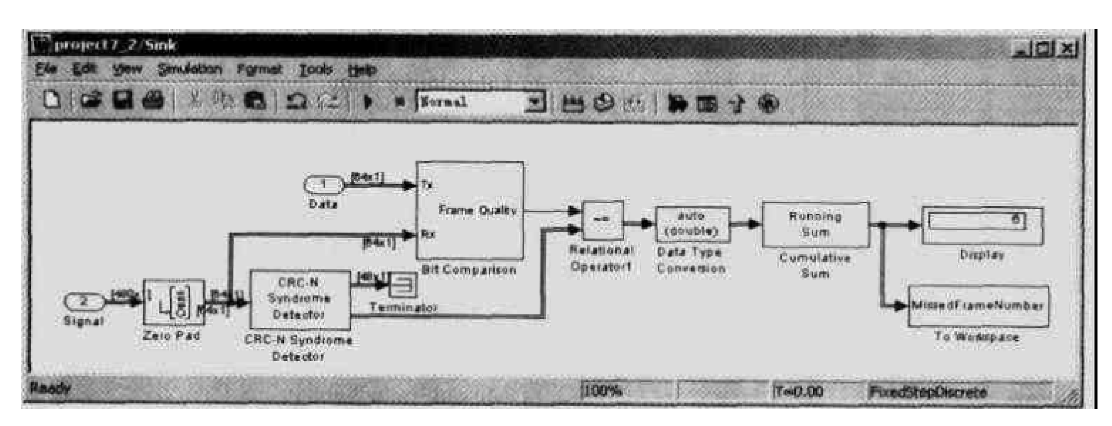



图 7-25　信宿模块的系统结构

信宿模块在接收到传输信号之后通过 Zero Pad（零填充模块）截取其中的 64 个比特用于 CRC 校验，这个校验过程由 CRC-N Syndrome Detector（CRC 检测模块）完成。CRC 检测器有两个输出信号，其中第一个输出端口的信号（即去除 CRC 校验位后的信息位）与编码前的信号一起进入 Bit Comparison（信号比较模块）。零填充模块和 CRC 检测模块的参数设置分别如表 7-17 和表 7-18 所示。

表 7-17　　**Zero Pad（零填充模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Zero Pad |
| Pad signal at | End |
| Pad along | Columns |
| Number of output rows | User-specified |
| Specified number of output rows | ProtectedDataWithCRC |
| Action when truncation occurs | None |

表 7-18　　**CRC-N Syndrome Detector（CRC 编码器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | CRC-N Syndrome Detector |
| CRC method | CRC-16 |
| Initial States | [0] |
| Checksums per frame | 1 |

信号比较模块的输出信号是关于48bit长度的数据帧的帧质量，当它等于0时，表示这48bit的数据帧没有出现传输错误；否则，如果出现了传输错误，信号比较模块输出1。同时，CRC检测模块的第二个输出端口的输出信号也是关于这个数据帧的校验结果，但它是根据CRC校验位进行判断的结果。如果CRC检验的结果与信号比较模块的输出信号相同，说明CRC校验是正确的；否则，CRC校验发生了错误的判决。Cumulative Sum（累加模块）对这两个信号的比较过程中结果不吻合的次数进行统计，统计结果一方面输出到Display（显示模块）中，另一方面通过To Workspace（工作区保存模块）保存到MATLAB工作区中名字为MissedFrameNumber的变量中。表7-19和表7-20所示是累加模块和工作区保存模块的参数设置。

**表 7-19　Cumulative Sum（累加模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Cumulative Sum |
| Sum input along | Channels (running sum) |
| Reset port | None |

**表 7-20　To Workspace（工作区保存模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | To Workspace |
| Variable name | MissedFrameNumber |
| Limit data points to last | inf |
| Decimation | 1 |
| Sample time (-1 for inherited) | -1 |
| Save format | Array |

图7-26所示是信号比较模块的系统结构框图。这两个信号通过Error Rate Calculation（误比特率统计模块）得到关于误比特率情况的统计数据，然后通过Selector（选择器模块）选择第二个输出信号，即误比特的个数作为输出。为了判断本帧数据是否出现了传输错误，我们把选择器模块的输出信号与它的一个单位延迟信号相减，如果它们的差为0，说明在本帧比较的过程中误比特数没有增加，因此本帧没有错误；否则，它们的差值就是本帧中的错误比特数。由于我们只需要知道本帧是否有错，通过Relational Operator（关系比较模块）后，如果数据帧没有错误，模块的输出信号等于0；否则，输出信号等于1。信号比较模块中的误比特率统计模块、选择器模块以及单位延迟模块的参数设置如表7-21～表7-23所示。

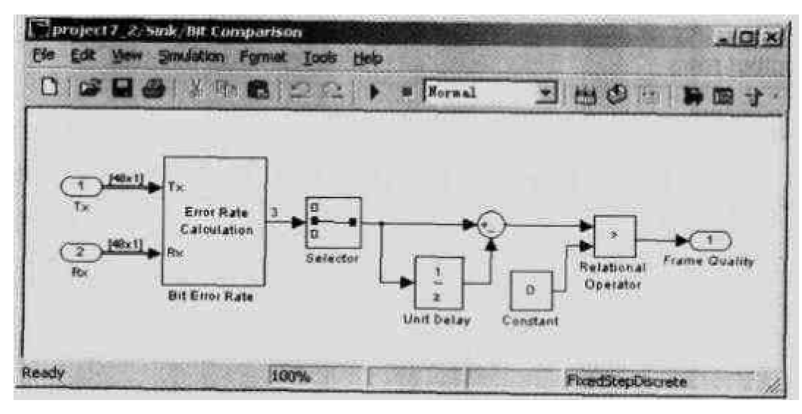



图 7-26　信号比较模块的系统结构

表 7-21　　**Error Rate Calculation**（误比特率统计模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Error Rate Calculation |
| Receive delay | 0 |
| Computation delay | 0 |
| Computation mode | Entire frame |
| Output data | Port |
| Reset port | Unchecked |
| Stop simulation | Unchecked |

表 7-22　　**Selector**（选择器模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Selector |
| Input Type | Vector |
| Source of element indices | Internal |
| Elements (-1 for all elements) | [2] |
| Input port width | 3 |

表 7-23　　**Unit Delay**（单位延迟模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Unit Delay |
| Initial conditions | 0 |
| Sample time (-1 for inherited) | -1 |

我们在 M 文件 project7_2main.m 中对 project7_2 中各个模块使用的变量的数值进行定义，完成图形绘制所需的其他操作，其代码如下：

```
% x 表示信道的误比特率
x=[0.00001 0.0001 0.001 0.005 0.01 0.02 0.03 0.04 0.05 0.1 0.2 0.3 0.4 0.5];
% y 表示发生错误判断的误帧所占的比例，它的长度与 x 相同
y=x;
% CRC 编码前的信息位的长度（bit）
ProtectedData = 48;
% 每帧数据之间的间隔（秒）
FrameInterval = 0.010;
% CRC 编码前的信号的符号周期
BitPeriod = FrameInterval/ProtectedData;
% CRC 编码之后码字的长度（bit）
ProtectedDataWithCRC = ProtectedData + 16;
% 每帧数据的长度（bit）
FrameLength = 480;
% 仿真时间（秒）
```

```
SimulationTime = 1000;
% 仿真时间内发送的帧的总数
TotalFrameNumber = SimulationTime/FrameInterval;
% 循环执行仿真程序
for i=1:length(x)
    % 信道的误比特率依次取 x 中的元素
    ChannelErrorRate=x(i);
    % 运行仿真程序，仿真结果保存在工作区变量 MissedFrameNumber 中
    sim('project7_2');
    % 计算 MissedFrameNumber 所占的比例
    y(i)=MissedFrameNumber(length(MissedFrameNumber))/TotalFrameNumber;
end
% 绘制 x 和 y 的关系曲线图，纵坐标采用对数坐标
loglog(x,y);
```

最后，在 MATLAB 工作区中输入命令行"project7_2main"，程序运行结束之后就得到如图 7-27 所示的对数曲线图。

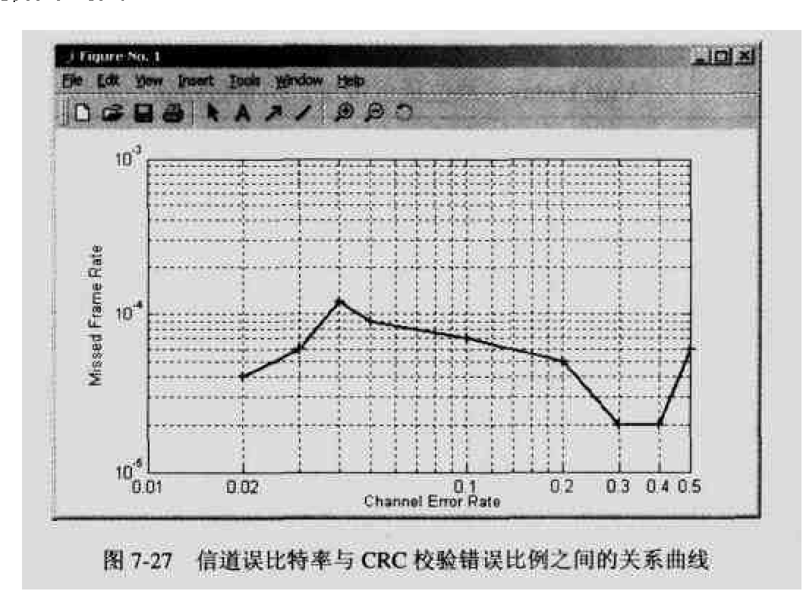



图 7-27　信道误比特率与 CRC 校验错误比例之间的关系曲线

从图 7-27 所示中可以看到，CRC 检测的性能是相当理想的，不管信道的误比特率是多高，CRC 检测器发生错误判决的比例都低于$10^{-4}$，即每 10000 个数据帧中只有一个帧在发生传输错误时未能被 CRC 检测器检查出来。因此，CRC 编码广泛的应用与移动通信系统中，用于实现自动请求重传（ARQ，Automatic Retransmission Request）功能。

## 7.3 卷积编码

卷积码与分组码不同。在分组码中，任何一段规定时间内编码器的输出完全取决于这段时间中的输入信息；而在卷积码中，任何一段规定时间内产生的 $n$ 个码元不仅取决于这段时间内的 $k$ 个信息位，而且还取决于前 $N$-1 段时间内的信息位，这个 $N$ 就称为卷积码的约束长度。

## 7.3.1 卷积编码器

在 MATLAB 中，卷积码既可以用多项式来表示，也可以用 Trellis 图来表示，其中后者具有更广的用途。本节我们将首先介绍卷积编码器的两种表示形式，在这个基础上介绍卷积编码器模块。

### 1. 卷积编码器的多项式表示

卷积编码器一般有一个或多个模二加法器，每个模二加法器都可以表示为一个多项式。对于带反馈的卷积编码器，它还有一个反馈多项式，用来表示卷积编码器的反馈连接方式。图 7-28 所示是一个具有一个输入端口和两个输出端口，并且具有两个移位寄存器的卷积编码器。

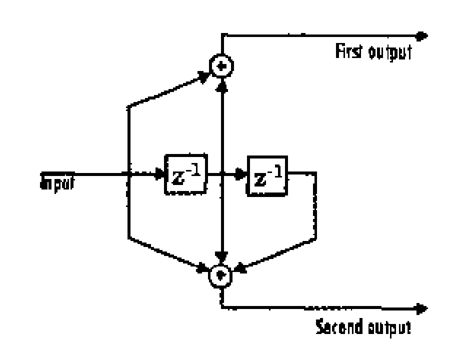



图 7-28　一个卷积编码的示例

卷积编码器的多项式表示由 3 部分组成：约束长度、生成多项式以及反馈连接多项式。当卷积编码器只有一个输入时，它的约束长度是一个标量，并且等于卷积编码器中储存的信息位的个数（包括移位寄存器的个数以及当前的输入信号）。例如，对于图 7-28 所示来说，卷积编码器的多项式的约束长度等于 3。如果卷积编码器有多个输入，则约束长度是一个向量，其中的每一个元素对应于一个输入信号在卷积编码器中储存的信息位的个数。图 7-29 所示是一个具有两个输入的卷积编码器，它的第一个输入信号在编码器中存储的长度是 5，第二个输入信号在编码器中储存的长度是 4，则这个编码器的约束长度是[5 4]。

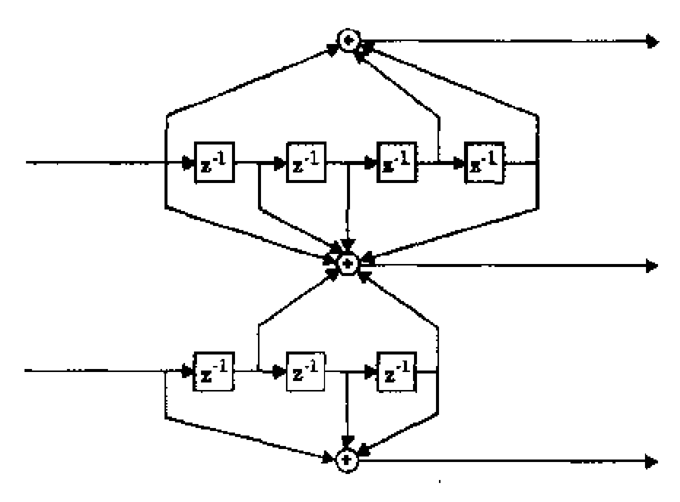


图 7-29　一个 2/3 卷积编码器的示例

假设卷积编码器有 $k$ 个输入信号和 $n$ 个输出信号，则这个卷积编码器的生成多项式是一个 $k$ 行 n 列的矩阵 $G=(g_{ij})_{k\times n}$，其中的每一个元素 $g_{ij}$ 表示第 $i$ 个输入信号对第 $j$ 个输出信号的影响：如果第 $i$ 个输入信号对第 $j$ 个输出信号有影响，则 $g_{ij}=1$；否则，$g_{ij}=0$。卷积编码器的生成多项式可以按照如下方式确定：对于每一个模二加法器，按照从左至右的顺序依次检查每个移位寄存器（包括当前的输入信号），如果这个寄存器与模二加法器之间有连接，则标记为 1，否则标记为 0，由此可以得到一个二进制序列。把这个二进制序列表示成八进制数后，就得到与这个模二加法器相对应的生成多项式。

对于图 7-28 所示中的卷积编码器，对应于第一个加法器的二进制序列是 110，对应于第二个加法器的二进制序列是 111，它们分别对应于八进制的 6 和 7，则这个卷积编码器的生成多项式是[6 7]。

在图 7-29 所示中，第一个模二加法器只与第一个输入信号有关，与之相对应的二进制序列是 10011。对于第二个模二加法器，它跟两个输入信号都有关系，其中与第一个输入信号相对应的二进制序列是 11101，与第二个输入信号相对应的二进制序列则是 0101。第三个模二加法器只与第二个输入信号有关，它的二进制序列是 1011。把二进制序列转化为八进制表示方式后，这个卷积编码器的生成多项式等于：

$$\begin{bmatrix} 23 & 35 & 0 \\ 0 & 5 & 13 \end{bmatrix}$$

对于带反馈的卷积编码器，还需要指定相应的反馈多项式。如果卷积编码器只有一个输入信号，则反馈多项式是一个标量。当卷积编码器有多个输入信号时，反馈多项式是一个向量，它的长度等于卷积编码器输入信号的个数。对于卷积编码器的一个输入信号，反馈多项式中的相应元素是一个八进制形式的二进制序列，其中的每个二进制位表示是否存在着从这个信号的某个移位寄存器的反馈连接。

图 7-30 所示是一个带反馈的卷积编码器，它的约束长度等于 5。对于第一个模二加法器，它与所有的移位寄存器都有连接，与之相对应的二进制序列是 11111，即八进制的 37；对于第二个模二加法器，它的二进制序列是 11011（八进制的 33），因此这个卷积编码器的生成多项式就等于[37 33]。在这个卷积编码器中，所有的移位寄存器都反馈回第一个模二加法器中，因此，它的反馈多项式等于 11111，转换成八进制数后等于 37。

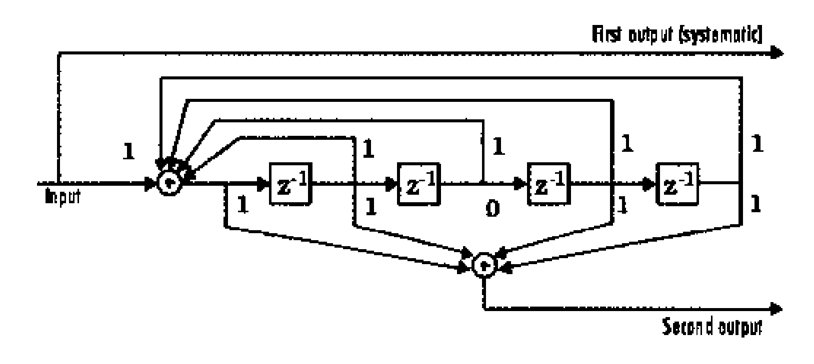



图 7-30　一个带反馈的卷积编码器

## 2．卷积编码器的 Trellis 表示

卷积编码器的 Trellis 表示方式直观地表达了卷积编码的输出信号与输入信号以及卷积编码器当前状态之间的关系。每个卷积编码器的多项式表示都可以转换成相应的 Trellis 表示形式，但是并非所有的 Trellis 表示形式都能表示为多项式。从这个意义上说，卷积编码器的 Trellis 图具有更广的应用范围，因此在 Simulink 中，卷积编码器和解码器的参数都是以 Trellis 图的方式表示。

图 7-31 所示是与图 7-28 所示的卷积编码器相对应的 Trellis 表示图。在这个 Trellis 表示图中，卷积编码器有 4 个状态（00，01，10 和 11），并且具有一位输入信号和两位输出信号。Trellis 表示图中的实线表示当前的输入信号是 0，虚线表示输入信号是 1，实线和虚线上的数字是输出信号的十进制表示。例如，假设卷积编码器当前状态是 01，当它的输入信号是 0 时，卷积编码器的输出信号是 1（即二进制的 01），同时转换到 00 状态；而当输入信号是 1 时，卷积编码器的输出信号是 2（即二进制的 10）。同时转换到 10 状态。

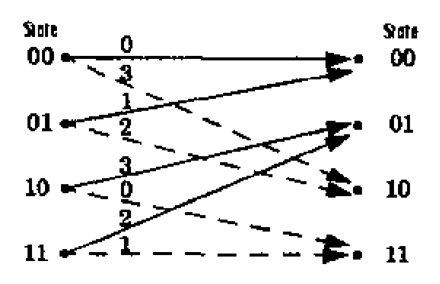



图 7-31　一个卷积编码的 Trellis 表示

在 MATLAB 中，一个具有 *k* 个输入和 *n* 个输出的卷积编码器的 Trellis 图是以 Trellis 结构表示的。一个 Trellis 结构由 5 个域组成，如表 7-24 所示。

表 7-24　Trellis 结构的 5 个域

| 域的名称 | 数据类型 | 说明 |
|---|---|---|
| numInputSymbols | 标量 | 卷积编码器输入信号的取值个数 $2^k$ |
| numOutputSymbols | 标量 | 卷积编码器输出信号的取值个数 $2^n$ |
| numStates | 标量 | 卷积编码器状态的个数 |
| NextStates | 具有 numStates 行 $2^k$ 列的矩阵 | 矩阵中的元素等于在指定的状态和输入信号条件下，卷积编码器的下一个状态 |
| outputs | 具有 numStates 行 $2^k$ 列的矩阵 | 矩阵中的元素等于在指定的状态和输入信号条件下，卷积编码器的输出信号 |

在 Trellis 结构中，输入信号、输出信号以及卷积编码器的状态都是用十进制数表示的。对于输入信号，第一个输入表示二进制序列的最高位；对于输出信号，第一个输出表示二进制序列的最高位。对于卷积编码器的状态，当 *k* 大于 1 时，第一个输入信号对应的移位寄存器表示二进制序列的最低位。

在 Trellis 结构中，*nextStates* 矩阵中的每个元素是介于 0 和 *numStates*-1 的一个整数，并且第 *i* 行第 *j* 列的元素表示在当前状态是 *i*-1，并且输入信号等于 *j*-1 时卷积编码器的下一个状态。例如，*nextStates* 矩阵中第二列的元素对应于输入信号是{0,…,0,1}时的转移状态。Trellis

结构中状态也是用十进制数表示的。

在 Trellis 结构的 outputs 矩阵中，第 *i* 行第 *j* 列的元素在表示当前状态是 *i*-1，并且输入信号等于 *j*-1 时卷积编码器的以十进制形式表示输出信号。下面的代码用于产生与图 7-31 所示相对应的 Trellis 结构。

```
>> trellis = struct('numInputSymbols',2,'numOutputSymbols',4,'numStates',4,...
'nextStates',[0 2;0 2;1 3;1 3],'outputs',[0 3;1 2;3 0;2 1])

trellis =
      numInputSymbols: 2
     numOutputSymbols: 4
            numStates: 4
           nextStates: [4x2 double]
              outputs: [4x2 double]
```

MATLAB 函数 trellis = poly2trellis(ConstraintLength, CodeGenerator, FeedbackConnection) 把多项式形式的卷积编码器转化成以 Trellis 结构表示的形式。例如，下面的代码把图 7-28 所示的卷积编码器的多项式转换成 Trellis 结构。从代码的运行结果中可以看到，这个 Trellis 结构与图 7-31 所示的内容是一致的。

```
>> trellis = poly2trellis(3,[6 7])

trellis =
      numInputSymbols: 2
     numOutputSymbols: 4
            numStates: 4
           nextStates: [4x2 double]
              outputs: [4x2 double]
>> trellis.nextStates
ans =
     0     2
     0     2
     1     3
     1     3
>> trellis.outputs
ans =
     0     3
     1     2
     3     0
     2     1
```

### 3．卷积编码器

MATLAB 中的卷积编码器（Convolutional Encoder）对二进制输入序列进行卷积编码，

得到相应的二进制输出序列。如果卷积编码器的输入信号的数目为 $k$，输出信号的数目为 $n$，则这个卷积编码器的码率为 $k/n$。卷积编码器模块及其参数设置对话框如图 7-32 所示。

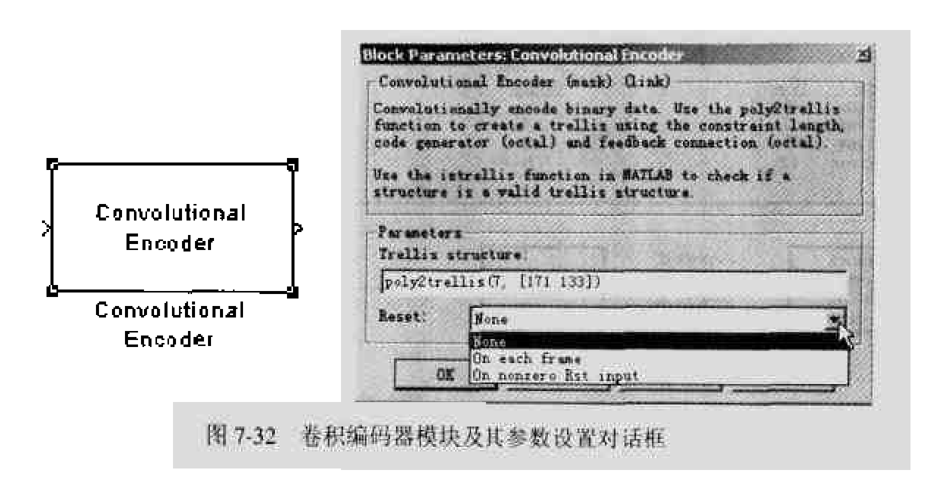



图 7-32　卷积编码器模块及其参数设置对话框

卷积编码器模块由两个参数。

■　Trellis structure（Trellis 结构）

卷积编码器的 Trellis 结构。通常在 MATLAB 工作区中设置一个 Trellis 结构的变量，在工作区中设置 Trellis 结构的各种参数，然后在卷积编码器模块的 Trellis structure 参数中填上该变量的名称。另外还可以通过 poly2trellis()函数把卷积编码器的约束长度、生成多项式以及反馈连接多项式转换成 Trellis 结构的形式。例如，当使用约束长度为 7，生成多项式为 171 和 133（八进制数），反馈连接多项式为 171（八进制数）的卷积编码器时，可以把本参数设置为 poly2trellis(7,[171 133],171)。

■　Reset（复位方式）

本参数用于确定卷积编码器的复位方式：None、On each frame 或 On nonzero Rst input。卷积编码器中的各种寄存器的初始状态都是 0。当复位方式设置为 None 时，卷积编码器在整个仿真过程中不对寄存器复位，这时候一般原始数据中包含了足够多的 0，这些 0 序列能够实现对寄存器复位。当复位方式设置为 On each frame 时，卷积编码器在每帧数据开始之前自动对寄存器复位。当复位方式设置为 On nonzero Rst input 时，卷积编码器增加一个输入端口 Rst，用于输入复位信号，并且在复位信号不等于零时对寄存器复位。

## 7.3.2　实例 7.3——IS-95 的卷积编码器

IS-95（Interim Standard-95）是由美国高通公司（Qualcomm）提出的一种 cdma 扩频通信标准，它跟 GSM 一样属于第二代移动通信技术的范畴，但是 IS-95 采用码分多址（Code Division Multiple Access）技术，而不是传统的时分多址（TDMA，Time Division Multiple Access）或频分多址（FDMA，Frequency Division Multiple Access），具有比较优越的语音通信质量。本节我们以 IS-95 中移动台的发射机为例介绍卷积编码器的应用。

IS-95 有两种速率集（Rate Set），其中速率集 1（RS-1）的传输速率可以是 9600bit/s、4800bit/s、2400bit/s 或 1200bit/s，而速率集 2（RS-2）的传输速率则分别等于 14400bit/s、7200bit/s、3600bit/s 和 1800bit/s，这两种速率集对应于两种不同的声码器。在此我们只介绍速率集 1。图 7-33 所示是 IS-95 速率集 1 移动台发射机的示意图。

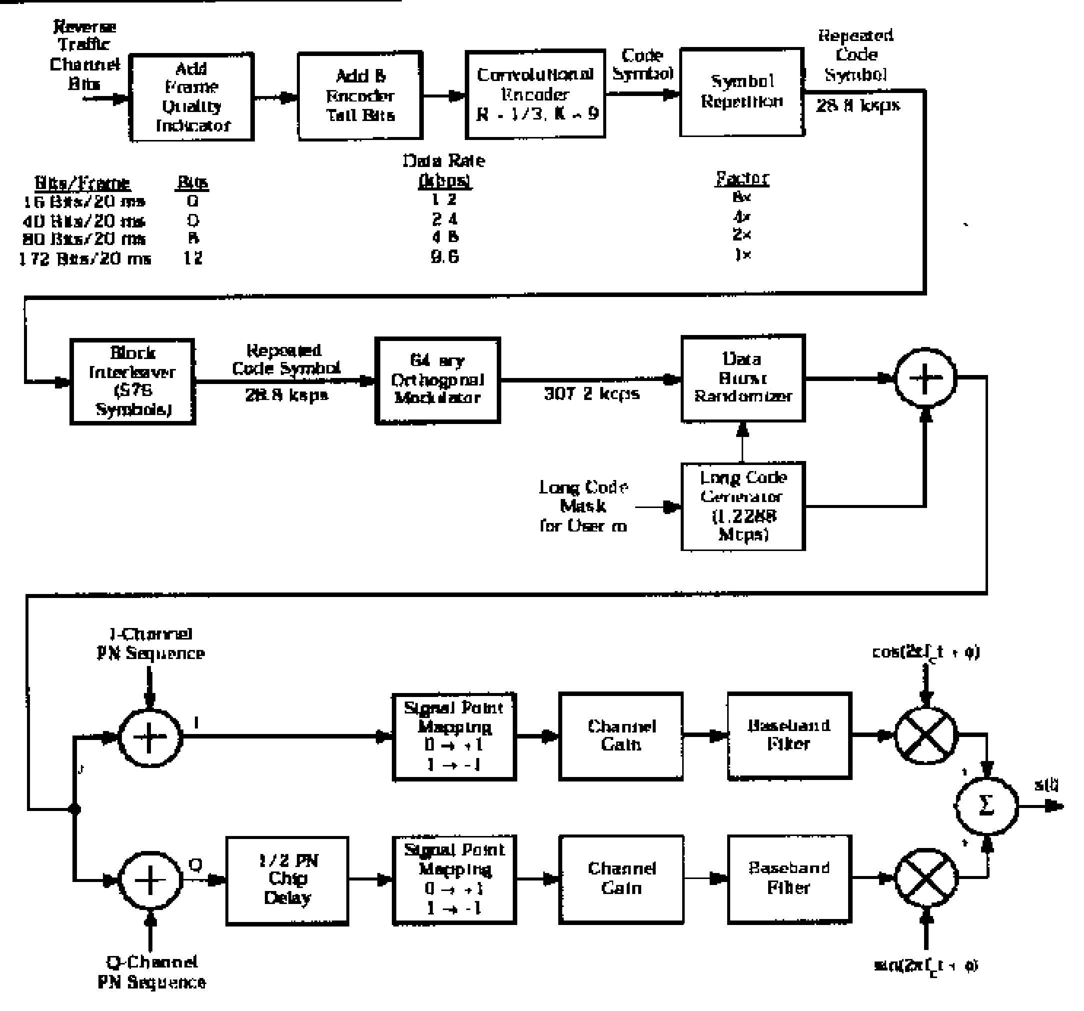



图 7-33　IS-95 速率集 1 移动台发射机示意图

IS-95 每帧的长度等于 20 毫秒，对应于速率集 1 的不同传输速率，移动台发射机接收的每帧数据的长度分别等于 172bit、80bit、40bit 和 16bit。这些数据首先通过一个 CRC 编码器，然后在末尾添加 8 个零（用于对卷积编码器复位），这时候每帧数据的长度分别等于 192bit、96bit、48bit 和 24bit，与之相应的数据传输速率是 9600bit/s、4800bit/s、2400bit/s、1200bit/s。对于速率集 1，IS-95 移动台发射机使用的卷积编码器的约束长度等于 9，码率等于 1/3，3 个生成多项式分别为 $g_0$=557，$g_1$=663，$g_2$=711，如图 7-34 所示。对于速率集 2，IS-95 移动台发射机使用的卷积编码器的码率等于 1/2。

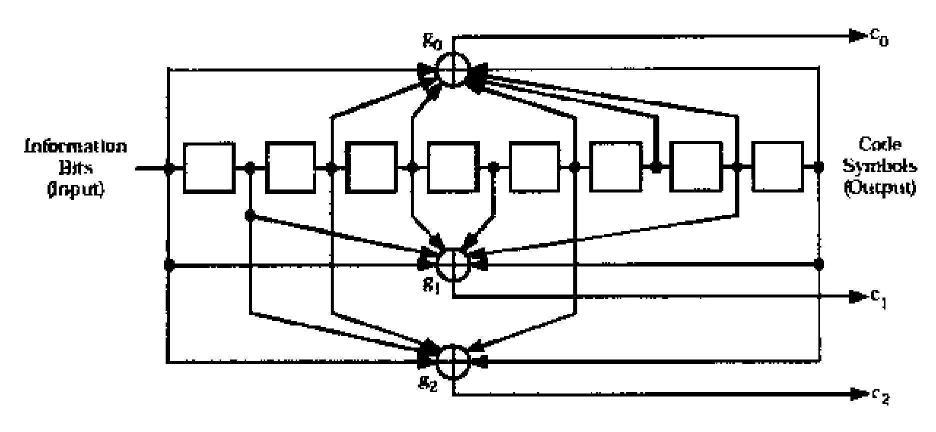



图 7-34　IS-95 码率为 1/3 的卷积编码器

通过卷积编码，IS-95 移动台发射机每帧数据的长度分别为 576bit、288bit、144bit 和 72bit。为了使不同速率的数据帧具有相同的长度，IS-95 通过 Repeat（信号重复）把这些数据帧的长度转换成 576（即传输速率为 28.8kbit/s）。本实例将对 IS-95 移动台发射机的 CRC 编码、卷积编码和信号重复过程进行仿真，其系统结构如图 7-35 所示。

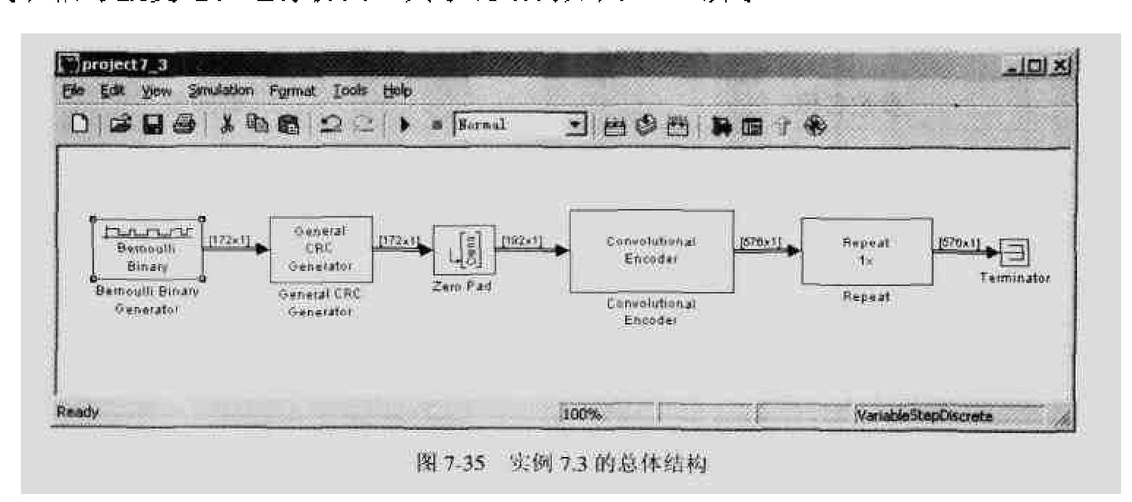



图 7-35　实例 7.3 的总体结构

在本实例中，数据源是一个 Bernoulli Binary Generator（贝努利二进制序列产生器），它每隔 20 毫秒产生一个数据帧，其中 0 和 1 出现的概率相同。贝努利二进制序列产生器的参数设置如表 7-25 所示。

**表 7-25　Bernoulli Binary Generator（贝努利二进制序列产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | 61 |
| Sample time | SampleTime |
| Frame-based outputs | Checked |
| Samples per frame | SamplesPerFrame |

根据不同的数据速率数据源产生不同长度的数据帧，这个长度由 SamplesPerFrame 参数确定（这些参数都将在 project7_3.m 文件中进行定义）。数据源产生的数据帧首先通过一个 CRC 产生器，产生一定长度的循环校验位。在 IS-95 中，不同的传输速率有不同长度的循环校验位：当传输速率等于 9600 bit/s 时，每帧数据的循环校验位长度等于 12bit，相对应的生成多项式是 $g(x)=x^{12}+x^{11}+x^{10}+x^{9}+x^{8}+x^{4}+x+1$；当传输速率等于 4800 bit/s 时，循环校验位的长度等于 8bit，生成多项式为 $g(x)=x^{8}+x^{7}+x^{4}+x^{3}+x+1$；而当传输速率等于 2400 或 1200 bit/s 时则不需要循环校验位。CRC 产生器的参数设置如表 7-26 所示。

**表 7-26　General CRC Generator（CRC 产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | General CRC Generator |
| Generator polynomial | CRCGeneratorPolynomial |
| Initial states | [0] |
| Checksums per frame | 1 |

在通过 CRC 产生器之后，每个数据帧还需要通过一个 Zero Pad（零填充模块）模块，在数据帧末端加入 8 个比特的 0，用于在每帧卷积编码结束之后对卷积编码器中的移位寄存器复位。由于 MATLAB 中的卷积编码器具有自动复位功能，因此这个零填充模块并不是必需的。表 7-27 和表 7-28 所示分别是零填充模块和卷积编码器的参数设置。

**表 7-27　Zero Pad（零填充模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Zero Pad |
| Pad signal at | End |
| Pad along | columns |
| Number of output rows | User-specified |
| Specified number of output rows | LengthWithEncoderTailBits |
| Action when truncation occurs | None |

**表 7-28　Convolutional Encoder（卷积编码器）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Convolutional Encoder |
| Trellis structure | ConvGenerator |
| Reset | None |

最后，卷积编码信号通过一个信号重复器（Repeat），使得每帧数据的长度都等于 576bit。对于速率集 1，当传输速率等于 9600bit/s 时，帧数据不需要重复（即重复次数等于 1）；当传输速率等于 4800bit/s、2400bit/s 和 1200bit/s 时，重复次数分别等于 2 次、4 次和 8 次。信号重复器的参数设置如表 7-29 所示。

**表 7-29　Repeat（信号重复器）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Repeat |
| Repetition count | RepetitionCount |
| Initial conditions | 0 |
| Frame-based mode | Maintain input frame size |

在本实例中我们定义了多个变量，这些变量的数值将在 project7_3main.m 中进行定义。下面的程序段是 project7_3main.m 文件的内容。

```
% 仿真时间设置为 1 秒
SimulationTime=1;
% 每帧的长度
FrameDuration=20/1000;
% 传输速率（9600/4800/2400/1200）
Rate=9600;
% 卷积编码器的生成多项式
ConvGenerator=poly2trellis(9, [557 663 711]);

switch Rate
```

```
    case 9600
        % Full Rate Parameters
        % 每帧抽样点的个数
        SamplesPerFrame=172;
        % CRC 校验的多项式
        CRCGeneratorPolynomial=[1 1 1 1 1 0 0 0 1 0 0 1 1];
        % 加入 8 位卷积编码尾后的帧数据的长度
        LengthWithEncoderTailBits=192;
        % 重复次数
        RepetitionCount=1;
    case 4800
        % Half Rate Parameters
        SamplesPerFrame=80;
        CRCGeneratorPolynomial=[1 1 0 0 1 1 0 1 1];
        LengthWithEncoderTailBits=96;
        RepetitionCount=2;
    case 2400
        % Fourth Rate Parameters
        SamplesPerFrame=40;
        CRCGeneratorPolynomial=[0];
        LengthWithEncoderTailBits=48;
        RepetitionCount=4;
    case 1200
        % Eighth Rate Parameters
        SamplesPerFrame=16;
        CRCGeneratorPolynomial=[0];
        LengthWithEncoderTailBits=24;
        RepetitionCount=8;
    otherwise
        error('Error: Parameter Rate should be 9600, 4800, 2400 or 1200!')
end

% 数据源产生数据的间隔
SampleTime=FrameDuration/SamplesPerFrame;
% 实施仿真
sim('project7_3');
```

要运行本实例程序，在 MATLAB 工作区中输入命令行“project7_3main”。如果需要对别的传输速率进行仿真，只需对文件 project7_3main.m 中相应参数的数值进行修改。本实例程序只对 IS-95 移动台发射机的前一部分的过程进行仿真，目的是介绍卷积编码器的使用方法，读者可以对本程序进行扩充以满足不同的要求。

### 7.3.3 卷积译码器

卷积译码器是卷积编码器的逆过程，用于从卷积编码器的输出信号中恢复出原始的信息

序列。MATLAB 中有两种类型的卷积译码器，即后验概率卷积译码器以及维特比译码器。本节我们将依次介绍这两种译码器。

1．后验概率译码器

后验概率译码器（APP Decoder）以后验概率 APP（Aposteriori Probability）的方式对卷积编码信号进行译码。后验概率译码器通常用于构建 Turbo 译码器。后验概率模块及其参数设置对话框如图 7-36 所示。

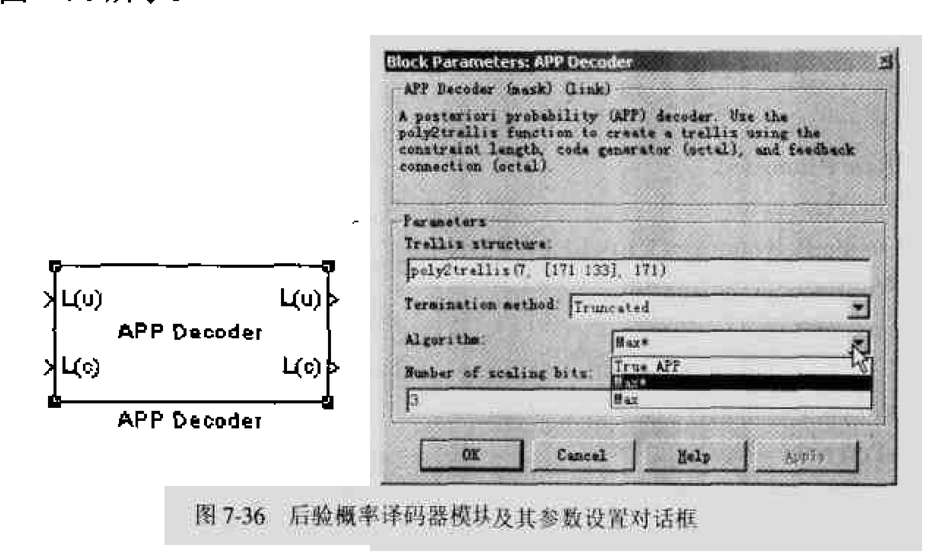


图 7-36　后验概率译码器模块及其参数设置对话框

后验概率译码器由两个输入信号序列：L(u)和 L(c)。输入信号 L(u)表示与解码器相对应的编码器的输入信号的对数似然概率，输入信号 L(c)则表示卷积编码信号的对数似然概率。后验概率译码器也有两个输出信号 L(u)和 L(c)，分别表示对输入信号序列 L(u)和 L(c)的纠正。如果卷积编码器输入信号的长度为 $k$，输出信号的长度为 $n$，则后验概率译码器的 L(c)向量的长度是 $n$ 的整数倍，L(u)向量的长度是 $k$ 的整数倍。当只需要输入信号 L(c)和输出信号 L(u)时，可以把输入信号端口 L(u)接地，即 L(u)的输入信号是 Ground 模块的输出信号，同时把输出端口 L(c)连接到 Terminator 模块。输出信号 L(u)通过一个硬判决器（Hard Decision）后得到原始的二进制信号序列。

后验概率译码器模块主要有以下几个参数。

■　Trellis structure（Trellis 结构）

与后验概率译码器相对应的卷积编码器的 Trellis 结构。它既可以是 MATLAB 工作区中的一个 Trellis 变量，也可以是通过 poly2trellis()函数产生的 Trellis 结构。

■　Termination method（编码器工作方式）

本参数用于确定与后验概率译码器相对应的卷积编码器在每帧数据的开始和结束阶段的工作方式：当设置为 Truncated 时，编码器在每帧数据开始之前对所有的移位寄存器进行复位；当设置为 Terminated 时，卷积编码器在到达全零状态时表示以帧数据的结束。如果卷积编码器的 Reset 参数等于 On each frame，则本参数应该设置为 Truncated。

■　Algorithm（译码算法）

后验概率译码器有 3 种译码算法：True APP、Max*和 Max。True APP 是一个纯粹的后验概率算法，而 Max*和 Max 都是后验概率的一种近似，它们具有比较高的运算速度。Max 选项采用 $\max\{a_i\}$作为近似值，而 Max*选项采用的是 $\max\{a_i\}$加上一个纠正值。当选择 Max*选项时，

Number of scaling bits 参数有效，用于设置后验概率译码器对数据进行调整的二进制位的数目。

■　Number of scaling bits（调整位的数目）

本参数用于确定在后验概率译码器调整数据的过程中涉及到的二进制位的个数，它是一个介于 0 和 8 之间的整数。本参数在 Algorithm 设置为 Max*时有效。

## 2．维特比译码器

图 7-37 所示是维特比译码器（Viterbi Decoder）模块，它通过维特比译码还原出二进制信号序列。如果卷积编码器输入信号的长度为 $k$，输出信号的长度为 $n$，则维特比译码器的输入信号和输出信号的长度分别是 $n$ 和 $k$ 的整数倍。

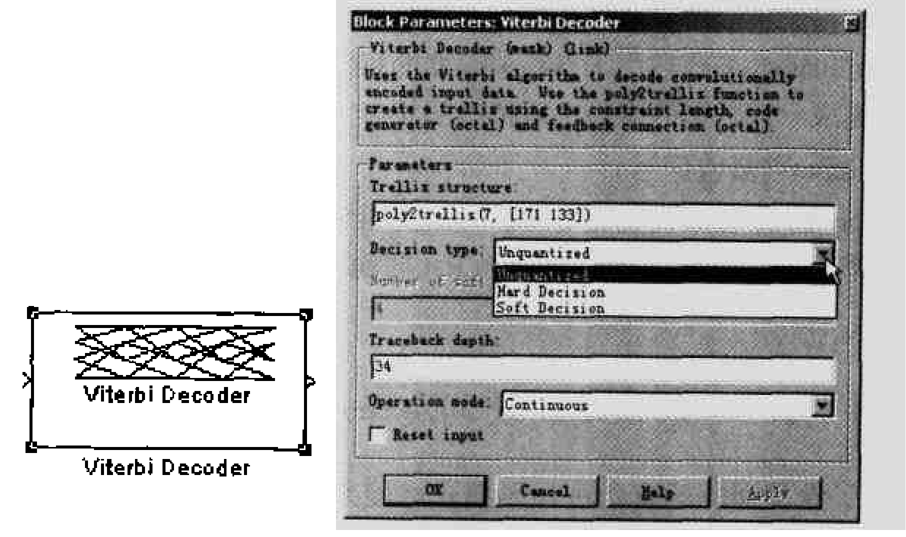



图 7-37　维特比译码器模块及其参数设置对话框

维特比译码器模块主要有以下几个参数。

■　Trellis structure（Trellis 结构）

与维特比译码器相对应的卷积编码器的 Trellis 结构。它既可以是 MATLAB 工作区中的一个 Trellis 变量，也可以是通过 poly2trellis()函数产生的 Trellis 结构。

■　Decision type（判决类型）

维特比译码器的判决类型有 3 种：Unquantized（非量化）、Hard Decision（硬判决）和 Soft Decision（软判决），如表 7-30 所示。

表 7-30　维特比译码器的判决类型

| 判决类型 | 解码器的输出类型 | 说明 |
|---|---|---|
| Unquantized | 实数 | +1 表示逻辑 0；−1 表示逻辑 1 |
| Hard Decision | 0，1 | 0 表示逻辑 0；1 表示逻辑 1 |
| Soft Decision | 介于 0 和 $2^b-1$ 之间的整数，其中 b 是软判决位的个数 | 0 表示具有取值为 0 的最大概率；$2^b-1$ 表示具有取值为 1 的最大概率；介于两者之间的数表示取 0 和 1 的相对概率。 |

■　Number of soft decision bits（软判决位的个数）

当 Decision type 设置为 Soft Decision 时，本参数有效，并且当它的取值为 $b$ 时，维特比

译码器的输出是介于 0 和 $2^b-1$ 之间的一个整数。

- Traceback depth（反馈深度）

反馈深度 D 影响着维特比译码的精度，同时也影响着解码的时延（即在输出第一个解码数据之前输出的 0 的个数）。

- Operation mode（操作模式）

维特比译码器有 3 种操作模式：Continuous、Terminated 或 Truncated。如果维特比译码器的输出信号是抽样信号，则应该把本参数设置为 Continuous 模式；当输入信号是帧数据时，操作模式可以是 Continuous、Terminated 或 Truncated。对于 Continuous 模式，维特比译码器在每帧数据结束时保存译码器的内部状态，用于对下一帧数据实施解码；在 Truncated 模式下，解码器在每帧数据结束的时候总能恢复到全零状态，它对应于卷积编码器的 On each frame 复位方式；Terminated 模式适用于卷积编码器的每帧输入信号的末尾有足够多的零，能够把卷积编码器在完成一帧数据的编码之后把内部状态复位为 0。

- Reset input（复位信号）

当 Operation mode 参数设置为 Continuous 并且选中了本选项前面的复选框之后，维特比译码器增加一个输入信号端口 Rst。同时当 Rst 的输入信号不等于 0 时，维特比译码器复位到初始状态。

## 7.3.4 实例 7.4——卷积码的软判决译码

前面提到了卷积译码器有两种译码模式：硬判决译码（hard decision）和软判决译码（soft decision），本实例我们介绍卷积译码器的软判决过程。图 7-38 所示是本实例的模块结构框图。

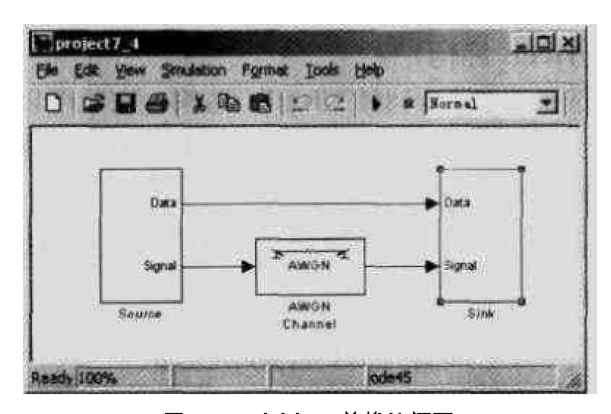



图 7-38 实例 7.4 的模块框图

本实例有 3 个子系统组成： Source（信源模块）对随机二进制信号进行卷积编码和二进制相位调制，输出基带调制信号；信道模块是一个 AWGN Channel（加性高斯白噪声信道）；Sink（信宿模块）对调制信号进行软判决译码，得到原始的信息序列，并且计算调制信号的误码率。在本实例程序中，我们将改变软判决的量化数目，研究量化间隔对软判决性能的影响。信源模块如图 7-39 所示。

图 7-39　信源模块的系统框图

信源模块由贝努利二进制序列产生器、卷积编码器以及二进制相位调制 3 个模块组成，它们的参数设置分别如表 7-31~表 7-33 所示。

**表 7-31　Bernoulli Binary Generator（贝努利二进制序列产生器）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | 25741 |
| Sample time | 0.0001 |
| Frame-based outputs | Checked |
| Samples per frame | 10000 |

**表 7-32　Convolutional Encoder（卷积编码器）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Convolutional Encoder |
| Trellis structure | poly2trellis(7, [171 133]) |
| Reset | On each frame |

**表 7-33　BIT/SK Modulator Baseband（二进制相位调制模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | BIT/SK Modulator Baseband |
| Phase offset (rad) | 0 |
| Samples per symbol | 1 |

本实例使用的信道是一个加性高斯白噪声信道，它在二相相位调制信号中叠加高斯白噪声。加性高斯白噪声模块的参数设置如表 7-34 所示。

**表 7-34　AWGN Channel（加性高斯白噪声模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | AWGN Channel |
| Initial seed | 1237 |
| Mode | Signal to noise ratio (SNR) |
| SNR (dB) | SNR |
| Input signal power (watts) | 1 |

信宿模块在接收到二进制相位调制信号后，首先由 Decision Maker（判决模块）对信号进行量化，得到软判决量化信号，然后通过 Viterbi Decoder（维特比译码器）对软判决信号实施译码。译码输出信号和信源模块产生的原始信号输入到 Error Rate Calculator（误比特率统计模块）中，统计得到的数据一方面通过 Display（显示模块）显示出来，另一方面通过一个 Selector（选择器）把其中的第一个元素（即编码信号的误比特率）保存到工作区变量 BitErrorRate 中。信宿模块如图 7-40 所示。

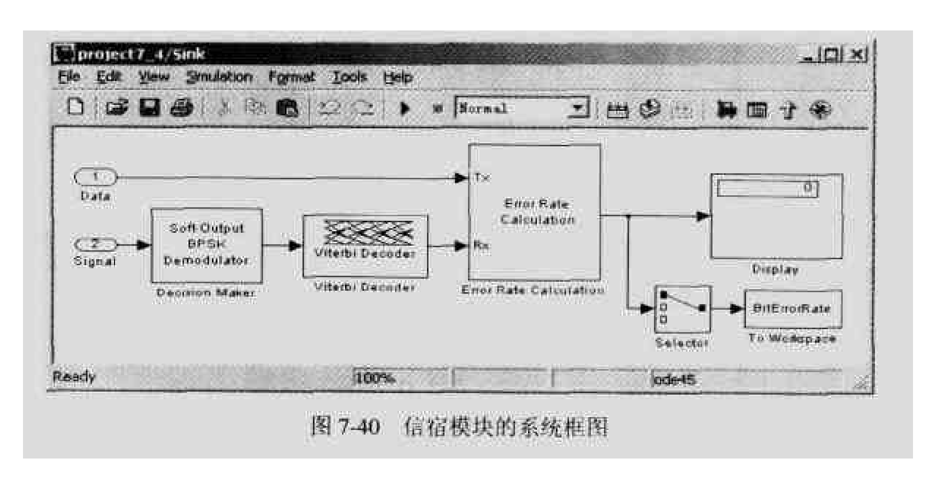



图 7-40 信宿模块的系统框图

图 7-41 所示是信宿模块中的 Decision Maker（判决模块）。在这个模块中，输入信号是复数形式的基带调制信号。对于二相相位调制（BIT/SK），信号 0 和 1 分别表示成$e^{j\theta}$ 和$e^{-j\theta}$，其中$\theta$ 是二相相位调制的相位偏移。在本实例中，BIT/SK 信号的相位偏移$\theta=0$，因此，调制信号分别等于±1，即调制信号的虚部等于 0。判决模块依据接收到的调制信号实部的强度进行量化，它通过 Standard Deviation（标准差模块）计算输入信号的标准差，然后用这个标准差除输入信号，并且通过增益等于－1 的 Gain（增益器）对信号进行翻转，得到介于±1之间的归一化数值。Uniform Encoder（均匀量化器）对归一化数值进行均匀量化，量化电平的数目由参数 DecisionBits 确定。由于维特比译码器在实施软判决译码时要求输入信号的类型是 double，而均匀量化器的输出信号是整型数据，因此需要进行数据类型转换，这个功能由 Data Type Conversion（数据类型转换模块）完成。

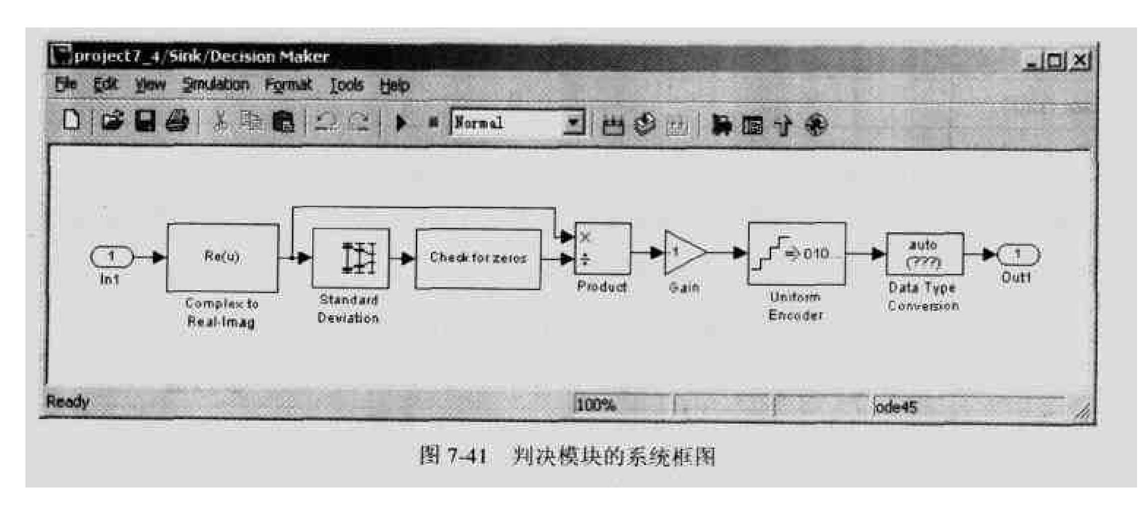



图 7-41 判决模块的系统框图

表 7-35、表 7-36 和表 7-37 列出了判决模块中的标准差模块、均匀量化器以及数据类型转换模块的参数设置。另外，在极端情形下，如果调制信号全等于 0，标准差模块计算得到的数值将等于 0，这是后执行除法将出错。为此，判决模块设置了一个 Check for zeros（零信号检测模块），

这个模块在输入信号为零时把输出信号设置为 1。图 7-42 所示是零信号检测模块的结构框图。

表 7-35　**Standard Deviation**（标准差模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Standard Deviation |
| Running standard deviation | Checked |
| Reset port | None |

表 7-36　**Uniform Encoder**（均匀量化器）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Uniform Encoder |
| Peak | 2 |
| Bits | DecisionBits |
| Output type | Unsigned integer |

表 7-37　**Data Type Conversion**（数据类型转换模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Data Type Conversion |
| Data type | auto |
| Saturate on integer overflow | Checked |

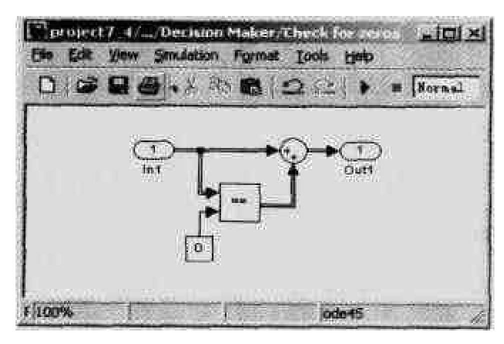


图 7-42　零信号检测模块的结构框图

当 DecisionBits 等于 $m$ 时，判决模块的输出信号是介于 0 和 $2^m-1$ 之间的整数，它们是维特比译码器软判决译码的输入信号。维特比译码器的参数设置如表 7-38 所示。

表 7-38　**Viterbi Decoder**（维特比译码器模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Viterbi Decoder |
| Trellis structure | poly2trellis(7, [171 133]) |
| Decision type | Soft Decision |
| Number of soft decision bits | DecisionBits |
| Traceback depth | 1000 |
| Operation mode | Truncated |

最后，为了对软判决译码和硬判决译码的性能进行比较，我们建立另外一个项目（命名

为 project7_4hard)，用于对相同的调制信号实施硬判决译码。在这个项目中，信源模块和信道模块都与上面介绍的 project7_4 的信源和信道相同，唯一的差别在于信宿模块，如图 7-43 所示。在这里，判决模块被 BIT/SK Demodulator Baseband（BIT/SK 模块）取代，同时 Viterbi Decoder（维特比译码器）将采用硬判决。BIT/SK 解调模块和维特比译码器的参数设置分别如表 7-39 和表 7-40 所示。

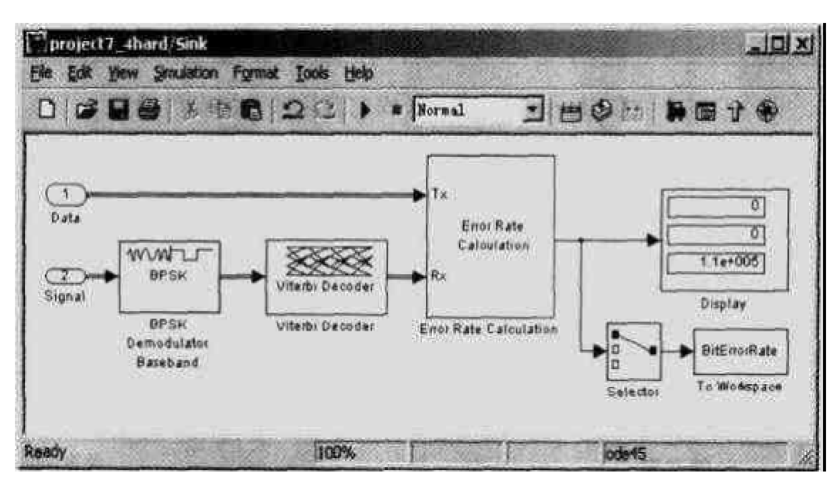



图 7-43 硬判决译码的信宿模块

表 7-39 BIT/SK Demodulator Baseband（BIT/SK 解调模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | BIT/SK Demodulator Baseband |
| Phase offset (rad) | 0 |
| Samples per symbol | 1 |

表 7-40 Viterbi Decoder（维特比译码器模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Viterbi Decoder |
| Trellis structure | poly2trellis(7, [171 133]) |
| Decision type | Hard Decision |
| Traceback depth | 1000 |
| Operation mode | Truncated |

M 文件 project7_4main.m 在不同的信噪比条件下重复执行前面建立的两个项目 project7_4 和 project7_4hard，然后绘制信道的信噪比与编码信号误比特率之间的关系曲线图，如图 7-44 所示。

```
% x 表示信噪比
x=-10:5;
% y 表示信号的误比特率，它的长度与 x 相同
y=x;
% 准备一个空白图形
hold off;

% 重复运行 project7_4，检验不同条件下软判决译码的性能
for index=2:4
```

```
    % 软判决的量化电平数
    DecisionBits=index;
    % 循环执行仿真程序
    for i=1:length(x)
        % 信道的信噪比依次取 x 中的元素
        SNR=x(i);
        % 运行仿真程序，得到的误比特率保存在工作区变量 BitErrorRate 中
        sim('project7_4');
        % 计算 BitErrorRate 的均值作为本次仿真的误比特率
        y(i)=mean(BitErrorRate);
    end
    % 绘制 x 和 y 的关系曲线图，纵坐标采用对数坐标
    semilogy(x,y);
    % 保持已经绘制的图形
    hold on;
end
% 重复运行 project7_4hard，检验不同条件下硬判决译码的性能
for i=1:length(x)
    % 信道的信噪比依次取 x 中的元素
    SNR=x(i);
    % 运行硬判决仿真程序，误比特率保存在工作区变量 BitErrorRate 中
    sim('project7_4hard');
    % 计算 BitErrorRate 的均值作为本次仿真的误比特率
    y(i)=mean(BitErrorRate);
end
% 绘制 x 和 y 的关系曲线图，纵坐标采用对数坐标
semilogy(x,y);
```

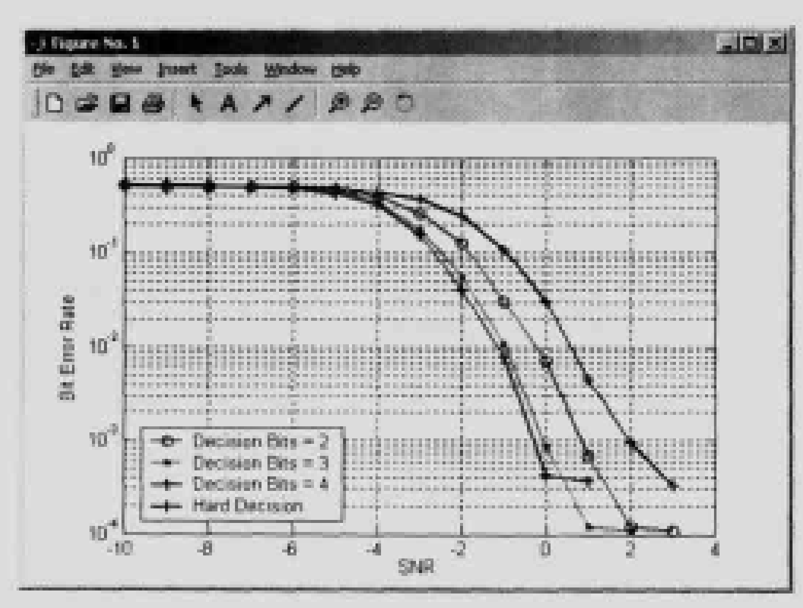



图 7-44　实例 7.4 的运行结果

图 7-44 所示中绘制的硬判决译码和量化电平分别等于 2、3、4 时软判决译码的误比特性能。从图中可以看到，对于卷积编码，软判决译码的误比特性能明显优于硬判决译码，并且软判决译码输入信号的量化电平数越多，译码后信号的误比特率越低。

# 7.4 块交织

块交织是一种基于分组的交织方法，它在一段时间内产生的交织信号与这段时间内的输入信号有关，它通过指定输入信号向量与输出信号向量下标之间的对应关系对输入信号进行置换。MATLAB 要求块交织器的输入信号和输出信号都是固定长度的向量，其中通用块交织器是其他类型的块交织器的基础。本节将依次介绍通用块交织、矩阵交织、代数交织以及随机交织。

## 7.4.1 通用块交织

通用块交织是把输入信号向量按照用户设定的顺序进行置换，产生交织信号的过程。通用块交织保证每个输入数据都能够出现在输出信号中，且每个输入信号只能出现一次。通用块交织是最基本的一种块交织方式，MATLAB 中的通用块交织器和通用块解交织器分别用于实现对信号的块交织和解交织过程。

### 1. 通用块交织器

通用块交织器（General Block Interleaver）按照指定的顺序对输入信号进行交织，产生相同长度的输出信号。通用块交织器的输入信号是一个长度为 $N$ 的向量 $X$（列向量或行向量），其中的每个元素 $X(i)$既可以是实信号，也可以是复信号。如果指定通用块交织器的置换方式向量为 $E$，则向量 $E$ 的长度等于 $N$，$E$ 中每个元素是介于 1 和 $N$ 之间的整数，并且输出向量 $Y$ 的第 $i$ 个元素 $Y(i)=X(E(i))$。需要注意的是，向量 $E$ 中的元素不应该出现重复。当输入信号是帧格式数据时，向量 $X$ 和 $E$ 都应该是列向量。通用块交织器及其参数设置对话框如图 7-45 所示。

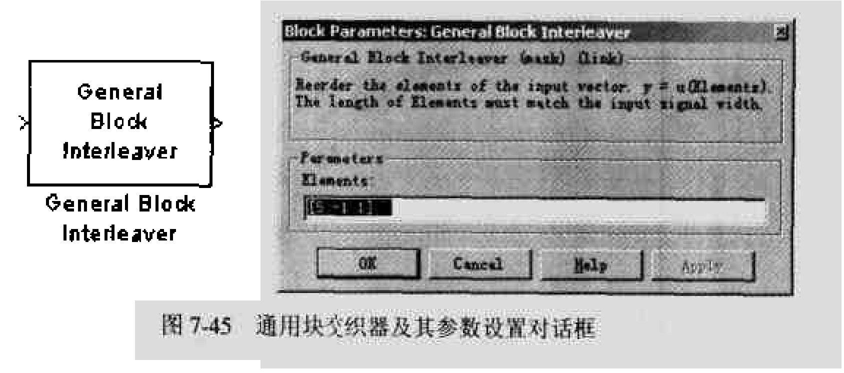



图 7-45 通用块交织器及其参数设置对话框

通用块交织器只有一个参数。

■ Elements（置换方式向量）

通用块交织器的置换方式向量 $E$，它的长度与输入信号的长度相等，并且 $E$ 中每个元素

是介于 1 和向量长度之间的整数。

### 2．通用块解交织器

通用块解交织器（General Block Deinterleaver）对交织信号实施解交织，还原得到交织之前的信号。假设通用块解交织器的输入信号是长度为 $N$ 的向量 $Y$（列向量或行向量），通用块解交织器的置换方式向量为 $D$，则输出向量 $Z$ 的第 $i$ 个元素 $Y(i)=Z(D(i))$。当输入信号是帧格式数据时，向量 $Y$ 和 $D$ 都应该是列向量。通用块解交织器及其参数设置对话框如图 7-46 所示。

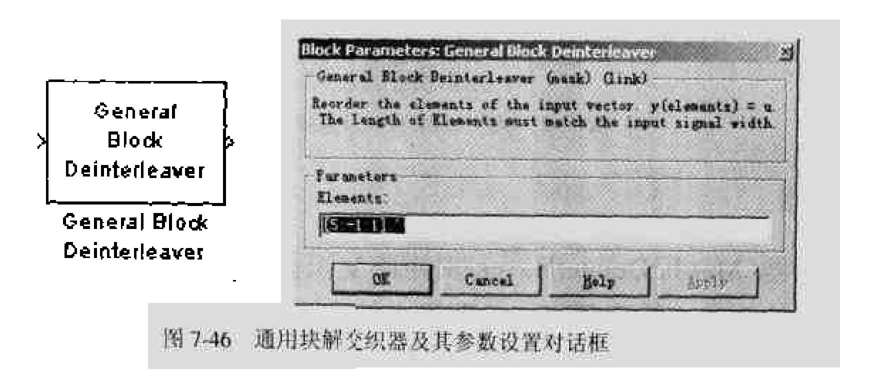



图 7-46　通用块解交织器及其参数设置对话框

在使用通用块解交织器对交织信号进行解调时，应该把置换方式向量 $D$ 设置为产生交织信号的通用块交织器中的置换方式向量 $E$ 相同。通用块解交织器有一个参数：

■　Elements（置换方式向量）

通用块解交织器的置换方式向量 $D$，它的长度与输入信号的长度相等，其中每个元素是介于 1 和向量长度之间的整数。

## 7.4.2　矩阵交织

矩阵交织是把输入信号按照某种顺序写入一个矩阵中，待完成整个矩阵的填充之后按照另外一种顺序从矩阵中读出数据的过程。MATLAB 提供了两种类型的模块来实现矩阵交织，即矩阵交织器和矩阵螺旋交织器，前者把输入信号按行写入矩阵，然后按列从矩阵中读出元素，后者则把输入信号按行写入矩阵，然后按照螺旋顺序从矩阵中读取出来。矩阵解交织器和矩阵螺旋解交织器则实现交织的逆过程，从相应的交织信号中还原出原始信号。

### 1．矩阵交织器

矩阵交织器（Matrix Interleaver）把输入信号按行写入矩阵，然后按列从矩阵中读出元素作为输出信号。矩阵交织器的输入信号是一个向量（行向量或列向量），假设矩阵 $M$ 是 $m$ 行 $n$ 列矩阵，则输入信号向量的长度等于 $m\times n$。当输入信号是帧格式数据时，输入信号向量是长度为 $m\times n$ 的列向量。矩阵交织器模块及其参数设置对话框如图 7-47 所示。

矩阵交织器模块有以下两个参数。

■　Number of rows（矩阵的行数）

矩阵交织器矩阵的行数 $m$。

■ Number of columns（矩阵的列数）

矩阵交织器矩阵的行数 $n$。

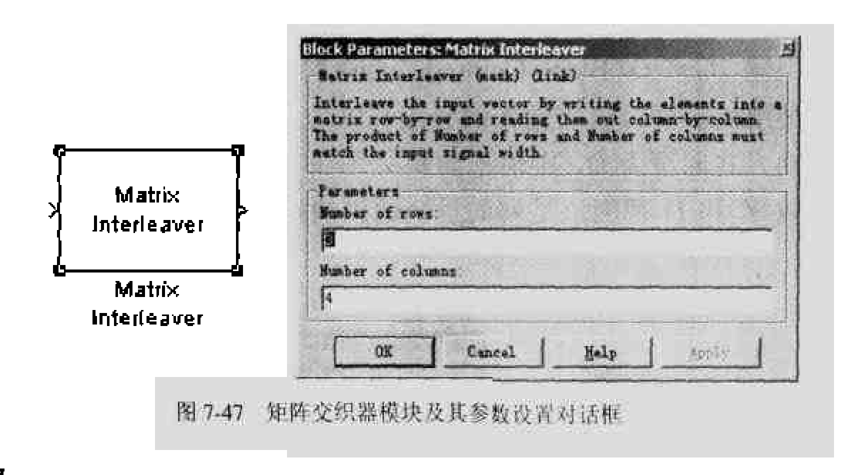



图 7-47 矩阵交织器模块及其参数设置对话框

## 2. 矩阵解交织器

矩阵解交织器（Matrix Deinterleaver）对矩阵交织器模块产生的交织信号实施解交织过程，还原得到交织前的信号。矩阵解交织器把输入的交织信号按列写入到 $m$ 行 $n$ 列矩阵 $M$ 中，然后按行依次读出矩阵 $M$ 的元素作为解交织信号。对比矩阵交织器的工作过程可以发现，这个过程实际上是矩阵交织器的逆过程。矩阵解交织器的输入信号是长度等于 $m\times n$ 的向量。当输入信号是帧格式数据时，输入信号向量是长度为 $m\times n$ 的列向量。矩阵解交织器模块及其参数设置对话框如图 7-48 所示。

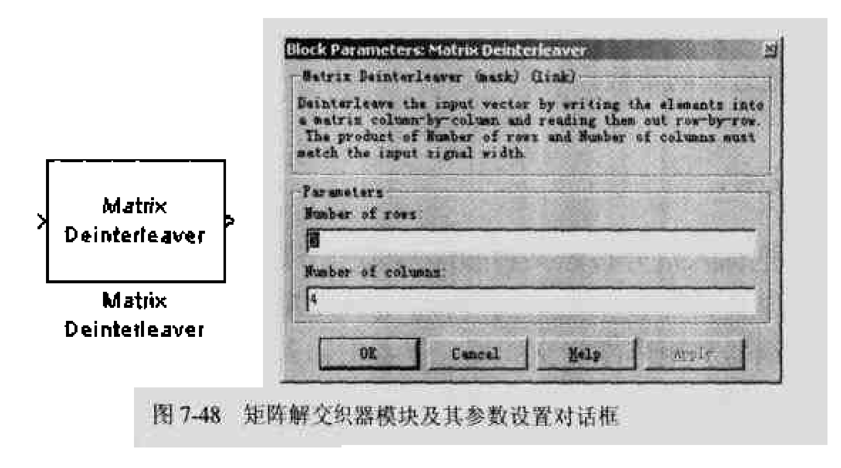



图 7-48 矩阵解交织器模块及其参数设置对话框

矩阵解交织器中矩阵 $M$ 的行数和列数应该与矩阵交织器矩阵的设置相同。矩阵解交织器模块有以下两个参数。

■ Number of rows（矩阵的行数）

矩阵解交织器矩阵的行数 $m$。

■ Number of columns（矩阵的列数）

矩阵解交织器矩阵的行数 $n$。

## 3. 矩阵螺旋交织器

矩阵螺旋交织器（Matrix Helical Scan Interleaver）把输入信号按行写入矩阵 $M$，然后按照螺旋顺序从矩阵中读出数据作为输出信号。假设矩阵螺旋交织器中矩阵 $M$ 是一个 $m$ 行 $n$

列矩阵，则输入信号是长度为 $m \times n$ 的向量（行向量或列向量）。当输入信号是帧格式数据时，输入信号应该是列向量。矩阵螺旋交织器模块及其参数设置对话框如图 7-49 所示。

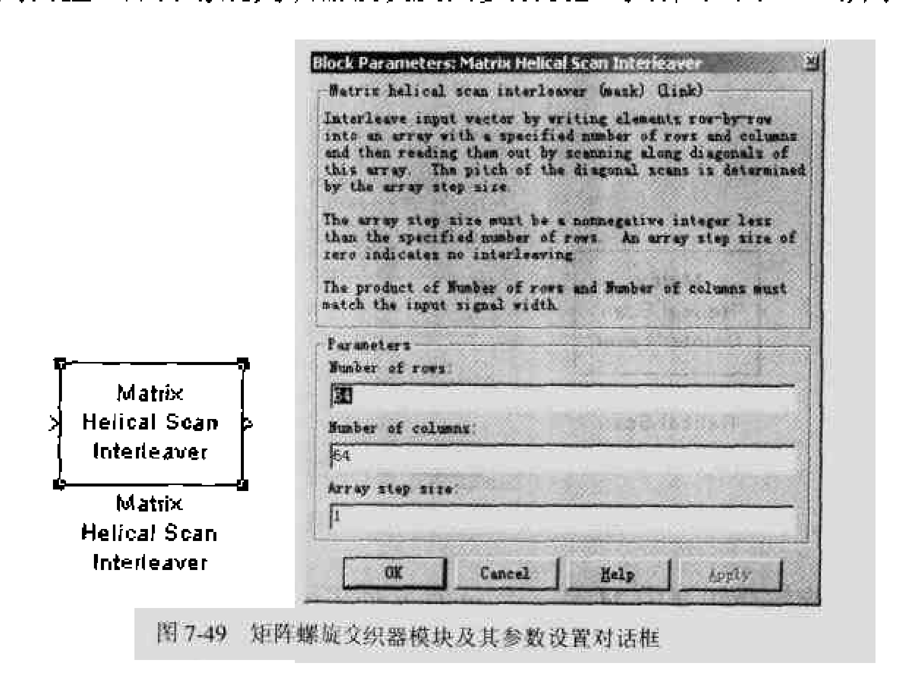



图 7-49　矩阵螺旋交织器模块及其参数设置对话框

矩阵螺旋交织器采用螺旋方式从矩阵 $M$ 中读取数据，即矩阵螺旋交织器首先按照斜线顺序读数据，然后下移一行，再依照斜线顺序读取数据，依次循环直到输出所有矩阵中的元素。具体说来，假设 $M$ 是一个 $m$ 行 $n$ 列矩阵，$M=(a_{i,j})_{m\times n}$，矩阵螺旋交织器的 Array step size 参数等于 $k$，则矩阵螺旋交织器首先读取以 $a_{1,1}$ 为起点的斜率为 $k$ 的斜线上的元素 $a_{1,1},a_{(1+k)\mathrm{mod}m,2},a_{(1+2k)\mathrm{mod}m,3},\cdots,a_{(1+(n-1)k)\mathrm{mod}m,n}$. 然后读取以 $a_{2,1}$ 为起点，斜率为 $k$ 的斜线上的元素 $a_{2,1},a_{(2+k)\mathrm{mod}m,2},a_{(2+2k)\mathrm{mod}m,3},\cdots,a_{(2+(n-1)k)\mathrm{mod}m,n}$，以此类推，最后读取以 $a_{m,1}$ 为起点，斜率为 $k$ 的斜线上的元素 $a_{m,1},a_{(m+k)\bmod m,2},a_{(m+2k)\bmod m,3},\cdots,a_{(m+(n-1)k)\bmod m,n}$，从而完成整个矩阵元素的输出。矩阵螺旋交织器有以下 3 个参数：

- **Number of rows（矩阵的行数）**

矩阵螺旋交织器中矩阵 $M$ 的行数 $m$。

- **Number of columns（矩阵的列数）**

矩阵螺旋交织器中矩阵 $M$ 的行数 $n$。

- **Array step size（斜率）**

矩阵螺旋交织器读取矩阵元素的斜率 $k$，它表示读取矩阵的过程中每增加一列时所增加的行数。当本参数设置为 0 时，矩阵螺旋交织器按行读取数据，这时候矩阵螺旋交织器没有对输入数据实施交织。

#### 4. 矩阵螺旋解交织器

矩阵解螺旋交织器（Matrix Helical Scan Deinterleaver）是矩阵螺旋交织器的逆过程，它把输入信号按照螺旋顺序写入矩阵 $M$，然后按行从矩阵中读出数据作为输出信号。假设矩阵螺旋交织器中矩阵 $M$ 是一个 $m$ 行 $n$ 列矩阵，则输入信号是长度为 $m \times n$ 的向量（行向量或列向量）。当输入信号是帧格式数据时，输入信号应该是列向量。矩阵螺旋解交织器模块及其参数设置对话框如图 7-50 所示。

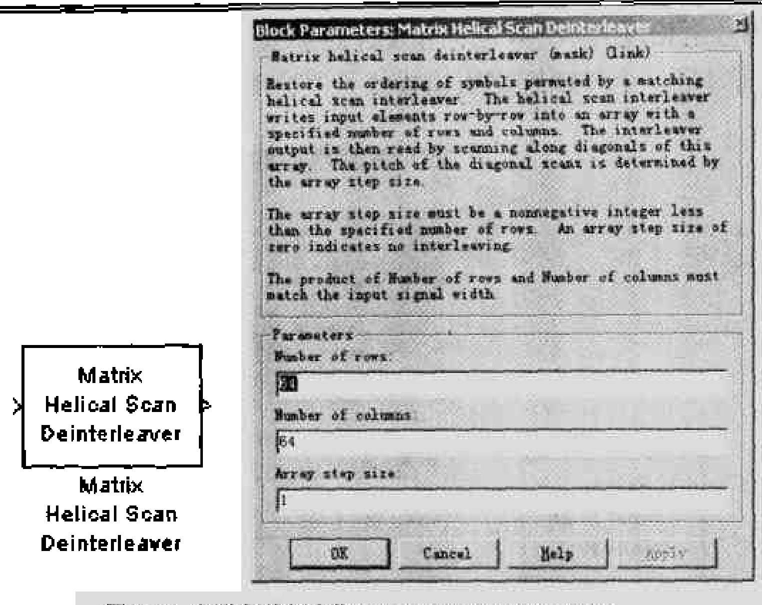



图 7-50 矩阵螺旋解交织器模块及其参数设置对话框

假设矩阵螺旋解交织器中 $M$ 是一个 $m$ 行 $n$ 列矩阵，$M=(a_{i,j})_{m\times n}$，斜率（Array step size）等于 $k$，则矩阵螺旋交织器首先把输入数据按照以 $a_{1,1}$ 为起点的，斜率为 $k$ 的斜线元素的顺序 $a_{1,1}, a_{(1+k)\bmod m,2}, a_{(1+2k)\bmod m,3}, \cdots, a_{(1+(n-1)k)\bmod m,n}$ 写入矩阵 $M$，然后把输入数据按照以 $a_{2,1}$ 为起点，斜率为 $k$ 的斜线元素的顺序 $a_{2,1}, a_{(2+k)\bmod m,2}, a_{(2+2k)\bmod m,3}, \cdots, a_{(2+(n-1)k)\bmod m,n}$ 写入矩阵，以此类推，最后把输入数据按照以 $a_{m,1}$ 为起点，斜率为 $k$ 的斜线上的元素 $a_{m,1}, a_{(m+k)\bmod m,2}, a_{(m+2k)\bmod m,3}, \cdots, a_{(m+(n-1)k)\bmod m,n}$ 顺序写入矩阵，从而完成整个矩阵元素的写入过程。矩阵螺旋解交织器有以下 3 个参数。

■ Number of rows（矩阵的行数）

矩阵螺旋解交织器中矩阵 $M$ 的行数 $M$。

■ Number of columns（矩阵的列数）

矩阵螺旋解交织器中矩阵 $M$ 的行数 $n$。

■ Array step size（斜率）

矩阵螺旋解交织器读取矩阵元素的斜率 $k$，它表示读取矩阵的过程中每增加一列时所增加的行数。

## 7.4.3 实例 7.5——交织器在 IS-95 中的应用

实例 7.3 介绍了 IS-95 移动台发射机的流程，并且编写了一个仿真模块，用于实现 IS-95 反向信道的 CRC 编码和卷积编码。本节将对这个模块进行扩充，在卷积编码的基础上实现信号的交织。

对于速率集 1，IS-95 反向业务信道的传输速率可以是 9600bit/s、4800bit/s、2400bit/s 以及 1200bit/s，每个 20 毫秒数据帧在通过 CRC 编码、卷积编码和重复之后长度都等于 576bit。这 576bit 数据按行输入到一个 32 行 18 列的矩阵中，然后按列读出数据作为交织信号。根据不同的速率 IS-95 规定了每列数据的读出顺序。对于 9600bit/s 的传输速率，交织器按照行号顺序读出一列数据，即对于某一列数据，首先读出这一列第一行一个元素，然后读出同一列

第二行的一个元素，以此类推，最后读出第 32 行的一个元素。对于 4800bit/s 的传输速率，读取每列数据的行号顺序是 1 3 2 4 5 7 6 8 9 11 10 12 13 15 14 16 17 19 18 20 21 23 22 24 25 27 26 28 29 31 30 32；对于 2400bit/s 的传输速率，这个顺序等于 1 5 2 6 3 7 4 8 9 13 10 14 11 15 12 16 17 21 18 22 19 23 20 24 25 29 26 30 27 31 28 32；而对于 1200 bit/s 的传输速率，这个顺序变成 1 9 2 10 3 11 4 12 5 13 6 14 7 15 8 16 17 25 18 26 19 27 20 28 21 29 22 30 23 31 24 32。本实例对实现 IS-95 移动台发射机速率集 1 的交织过程进行仿真。

本实例采用与实例 7.3 类似的 Source Coding（信源编码模块）对产生的随机数据帧实施 CRC 编码和卷积编码，然后通过一个 Matrix Interleaver（矩阵交织器）对卷积信号实施矩阵交织，然后采用 General Bolck Interleaver（通用块交织模块）再次对矩阵交织信号实施交织，得到 IS-95 移动台的交织信号。实例 7.5 的系统结构如图 7-51 所示。

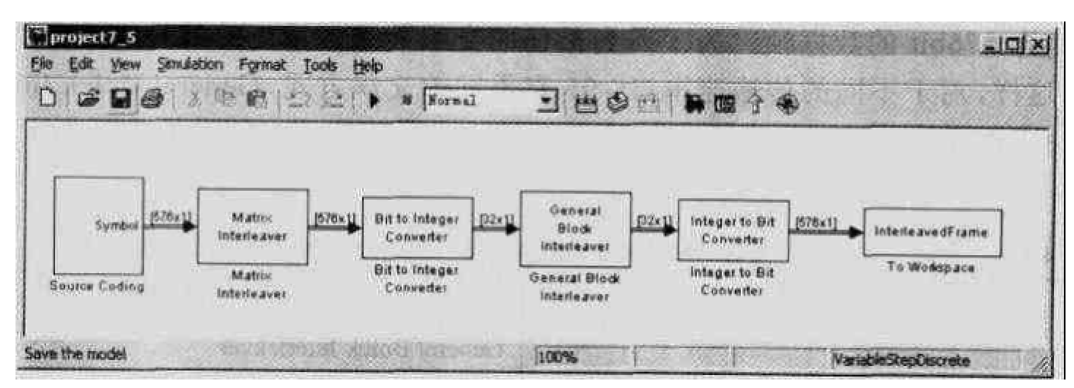



图 7-51　实例 7.5 的系统结构

图 7-52 所示是信源编码模块的系统结构。在信源编码模块中，Bernoulli Binary Generator（贝努利二进制序列产生器）根据不同的传输速率产生不同长度的二进制数据帧，这些数据帧首先通过通用 General CRC Generator（CRC 编码器）产生一个 CRC 校验，然后通过 Convolutional Encoder（卷积编码器）对信号实施卷积编码，最后信号进入 Repeat（重复模块），使得不同速率的数据帧产生相同长度的编码信号。另外，为了使卷积编码器中的寄存器在每帧数据的编码过程结束之后能够恢复到零状态，CRC 编码后的数据帧通过 Zero Pad（零填充模块）在信息位后面添加 8bit 的零信号，然后进入卷积编码器进行卷积编码。信源编码模块中各个子模块的参数设置与实例 7.3 中的相应模块的参数设置相同。

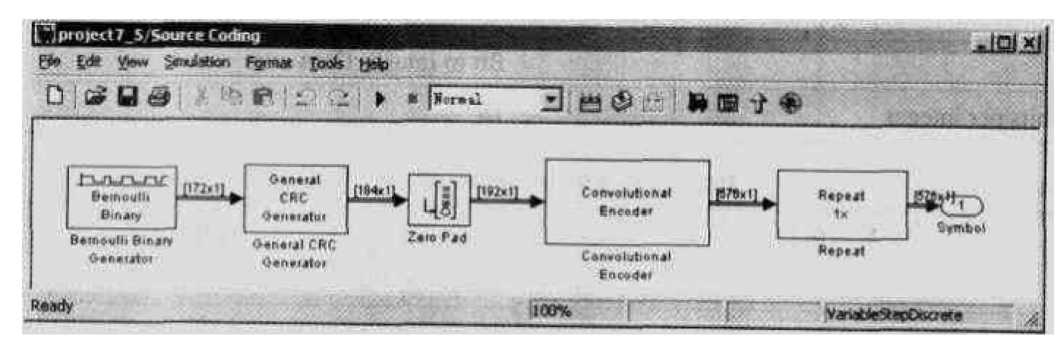



图 7-52　信源编码模块的系统结构

信源编码模块的输出信号是一个长度为 576bit 的数据帧，这些数据首先进入一个 32 行 18 列的矩阵交织器（Matrix Interleaver）进行交织。这个矩阵交织器把输入的二进制数据按照行顺序依次写入矩阵，然后按照列顺序从矩阵中读出。矩阵交织器的参数设置如表 7-41 所示。

表 7-41　　**Matrix Interleaver（矩阵交织器）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Matrix Interleaver |
| Number of rows | 32 |
| Number of columns | 18 |

在 IS-95 中，除全速（9600 bit/s）数据帧之外，其他数据帧的交织并不是纯粹的矩阵交织，这些数据帧按行写入矩阵，然后按列顺序读出数据，但是在读取一列数据的时候，不同速率的数据帧有不同的顺序。造成这种处理方式的原因在于，卷积编码信号通过信号重复之后，相邻两个比特可能是相同的重复数据。例如，对于 1200 bit/s 的传输速率，它经过卷积编码之后只有 72bit，其中每个比特符号需要重复 7 次，从而得到长度为 576bit 的数据帧。为了得到最佳的交织效果，重复的信号应该有尽可能大的间隔，这样，对于不同的传输速率，IS-95 有不同的交织方式。General Bolck Interleaver（通用块交织模块）对矩阵交织器的输出信号再次实施交织，其参数设置如表 7-42 所示。

表 7-42　　**General Bolck Interleaver（通用块交织模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | General Bolck Interleaver |
| Elements | InterleaverElements |

矩阵交织器的输出信号是一个长度为 576bit 的向量，它可以看作是一个 32 行 18 列的矩阵，因此我们把每一行元素转换成一个 18bit 的整数，从而得到一个长度为 32 的整型列向量。通用块交织模块按照指定的顺序对这个列向量实施交织，交织信号再通过一个相反的变换过程（即把每一个整数转换成 18bit 的二进制向量），其效果等同于按照特定的顺序从矩阵中读出元素。

Bit to Integer Converter 模块和 Integer to Bit Converter 模块分别把二进制向量转换成整数和把整数转换成一个二进制向量，其参数设置分别如表 7-43 和表 7-44 所示。

表 7-43　　**Bit to Integer Converter 模块的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Bit to Integer Converter |
| Number of bits per integer | 18 |

表 7-44　　**Integer to Bit Converter 模块的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Integer to Bit Converter |
| Number of bits per integer | 18 |

最后，通过 To Workspace（工作区写入模块）把交织信号保存在工作区变量 InterleavedFrame 中。表 7-45 所示是工作区写入模块的参数设置情况。

表 7-45　　To Workspace（工作区写入模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | To Workspace |
| Variable name | InterleavedFrame |
| Limit data points to last | inf |
| Decimation | 1 |
| Sample time (-1 for inherited) | -1 |
| Save format | Array |

M 文件 project7_5main.m 对 project7_5 中使用的各种变量进行定义，其程序代码如下：

```
% 仿真时间设置为 1 秒
SimulationTime=1;
% 每帧的长度
FrameDuration=20/1000;
% 传输速率（9600/4800/2400/1200）
Rate=9600;
% 卷积编码器的生成多项式
ConvGenerator=poly2trellis(9, [557 663 711]);

switch Rate
    case 9600
        % Full Rate Parameters
        % 每帧抽样点的个数
        SamplesPerFrame=172;
        % CRC 校验的多项式
        CRCGeneratorPolynomial=[1 1 1 1 1 0 0 0 1 0 0 1 1];
        % 加入 8 位卷积编码尾后的帧数据的长度
        LengthWithEncoderTailBits=192;
        % 重复次数
        RepetitionCount=1;
        % 行交织方式
        InterleaverElements=[1:32]';
    case 4800
        % Half Rate Parameters
        SamplesPerFrame=80;
        CRCGeneratorPolynomial=[1 1 0 0 1 1 0 1 1];
        LengthWithEncoderTailBits=96;
        RepetitionCount=2;
        InterleaverElements=[1 3 2 4 5 7 6 8 9 11 10 12 13 15 14 16 17 19 18 20 21
```

```
                    23 22 24 25 27 26 28 29 31 30 32]';
    case 2400
        % Fourth Rate Parameters
        SamplesPerFrame=40;
        CRCGeneratorPolynomial=[0];
        LengthWithEncoderTailBits=48;
        RepetitionCount=4;
        InterleaverElements=[1 5 2 6 3 7 4 8 9 13 10 14 11 15 12 16 17 21 18 22 19
                    23 20 24 25 29 26 30 27 31 28 32]';
    case 1200
        % Eighth Rate Parameters
        SamplesPerFrame=16;
        CRCGeneratorPolynomial=[0];
        LengthWithEncoderTailBits=24;
        RepetitionCount=8;
        InterleaverElements=[1 9 2 10 3 11 4 12 5 13 6 14 7 15 8 16 17 25 18 26 19
                    27 20 28 21 29 22 30 23 31 24 32]';
    otherwise
        error('Error: Parameter Rate should be 9600, 4800, 2400 or 1200!')
end

% 数据源产生数据的间隔
BitPeriod=FrameDuration/SamplesPerFrame;

% 打开模块 project7_5
project7_5;
% 实施仿真
sim('project7_5');
```

要运行仿真，只需在工作区中输入命令行“project7_5main”。本实例只是对 IS-95 移动台发射机使用的交织算法进行介绍，读者可以根据自己的设计需要对这个实例的功能进行扩充，实现各种仿真功能。

## 7.4.4 代数交织

代数交织是通过某种代数变换算法产生一个关于下标的置换向量，然后根据置换向量对输入信号进行交织的过程。MATLAB 提供了两种不同的代数交织算法，即 Takeshita-Costello 算法和 Welch-Costas 算法，它们对输入信号的长度由不同的要求，同时产生不同的交织信号。本届我们将依次介绍关于这两种算法的交织器和解交织器。

1．代数交织器

代数交织器（Algebraic Interleaver）允许用户用两种代数方法（Takeshita-Costello 或 Welch-Costas）对输入信号进行交织。代数交织器的输入信号是一个长度为 $N$ 的向量 $X$。如果输入信号是抽样信号，它可以是列向量或行向量；否则，如果输入信号是帧信号，它必须是列向量。代数交织器模块及其参数设置对话框如图 7-53 所示。

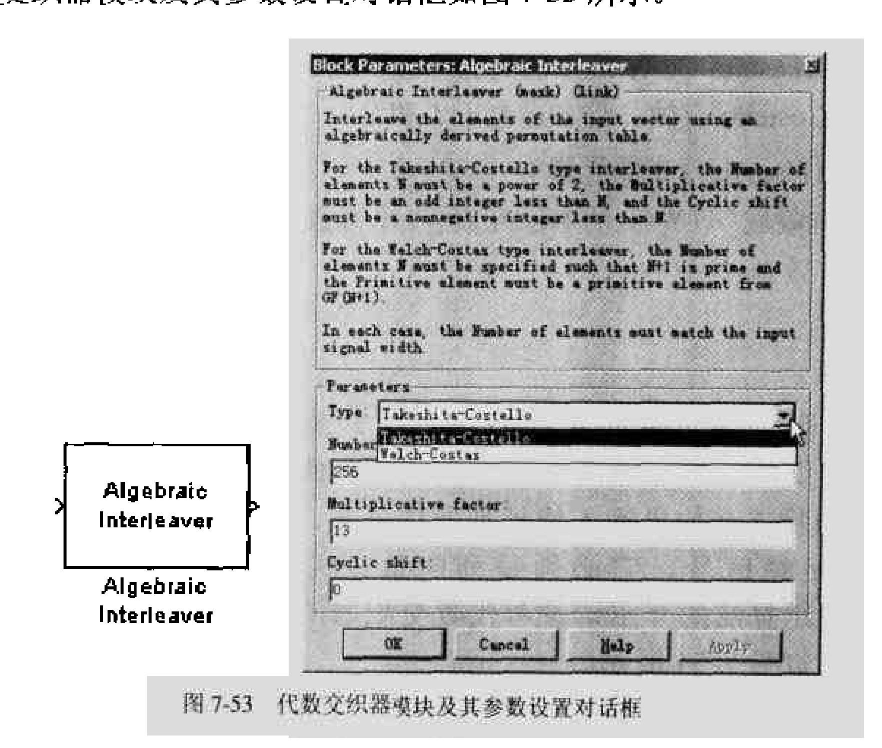



图 7-53 代数交织器模块及其参数设置对话框

当代数交织器的交织方法设置为 Welch-Costas 时，它通过有限域 GF($N+1$) 对输入信号进行交织。这时候要求 $N+1$ 是一个素数，如果选取一个小于 $N$ 且与 $N$ 互质正整数 $\alpha$ 作为有限域 GF($N+1$) 的本原元素，则这个有限域中的非零元素都可以表示成本原元素 $\alpha$ 的幂。Welch-Costas 算法把第 $i$ 个输入信号 $X_i$ 映射为第 $j$ 个输出信号 $Y_j$=$X_i$，其中下标 $i$ 和 $j$ 满足关系 $j=\alpha^{i-1}\bmod(N+1)$。

例如，当 $N=6$ 时，有限域 GF(7) 的本原元素 $\alpha$ 等于 3 或 5。取 $\alpha=3$，则根据公式 $j=\alpha^{i-1}\bmod(N+1)=3^{i-1}\bmod 7$，输入信号的下标[1,2,3,4,5,6]被映射为[1,3,2,6,4,5]，即输出向量的第一个元素等于输入向量的第一个元素，输出向量的第二个元素等于输入向量的第三个元素，等等。

当代数交织器的交织方法设置为 Takeshita-Costello 时，输入信号向量的长度 $N=2^m$，$m\in\mathrm{Z}$。Takeshita-Costello 交织算法首先构建一个长度为 $N$ 循环向量 $V$，向量中每个元素 $V(i)=(k\times(i-1)\times i/2)\mathrm{mod}N$，其中参数 $k$ 是 Takeshita-Costello 算法的乘法因子，且 $k\in\mathrm{Z}$，$0<k<N$。Takeshita-Costello 算法根据循环向量 $V$ 构造置换向量 $P$，然后把置换向量循环左移 $d$ 位（$d$ 称为循环移位数），得到代数交织器的输出信号。

代数交织器有以下几个参数。

■ Type（交织算法）

代数交织器的交织算法，它可以设置为 Takeshita-Costello 或 Welch-Costas。

■ Number of elements（输入向量的长度）

代数交织器输入信号向量的长度 $N$。

■ Multiplicative factor（乘法因子）

代数交织器 Takeshita-Costello 算法的乘法因子 $k$。本参数在代数交织器的交织算法 Type 设置为 Takeshita-Costello 有效。

■ Cyclic shift（循环移位数）

代数交织器 Takeshita-Costello 算法的循环移位数 $d$ 本参数在代数交织器的交织算法 Type 设置为 Takeshita-Costello 有效，且 $d$ 是介于 1 和 $N-1$ 之间的整数。

■ Primitive element（本原元素）

代数交织器有限域 $GF(N+1)$ 的本原元素 $\alpha$。本参数在代数交织器的交织算法 Type 设置为 Welch-Costas 时有效。

### 2. 代数解交织器

代数交织器（Algebraic Deinterleaver）对 Takeshita-Costello 交织信号或 Welch-Costas 交织信号进行解交织，得到交织前的信号。代数交织器的输入信号是一个长度为 $N$ 的向量 $X$。如果输入信号是抽样信号，它可以是列向量或行向量；否则，如果输入信号是帧信号，它必须是列向量。关于 Takeshita-Costello 交织算法和 Welch-Costas 交织算法的介绍请参考代数交织器的相关内容。代数解交织器模块及其参数设置对话框如图 7-54 所示。

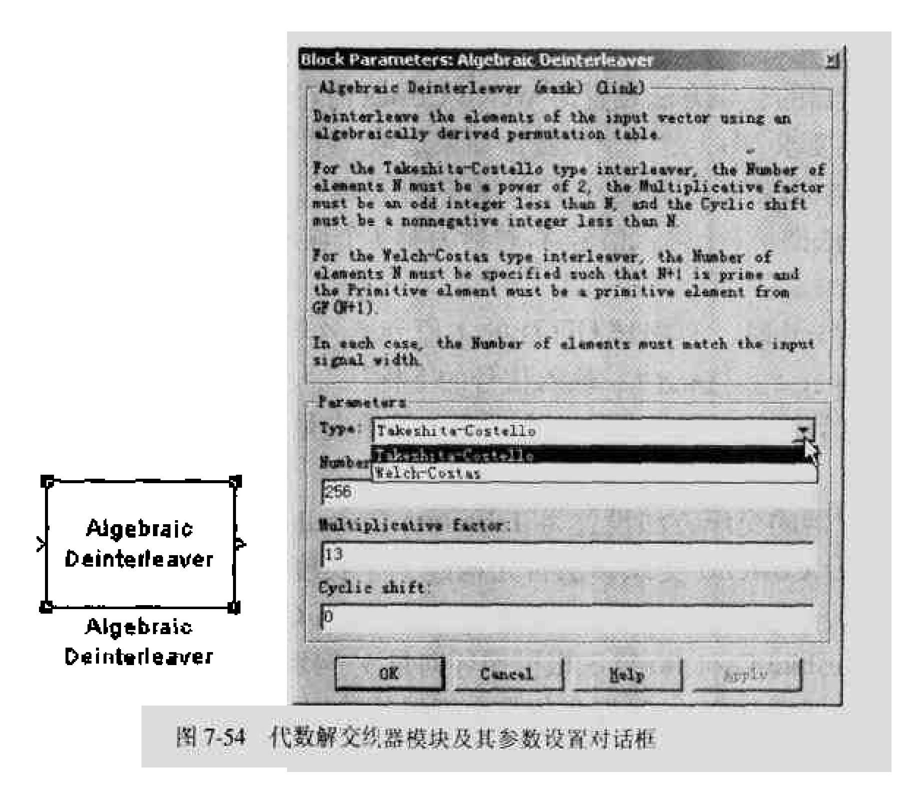


图 7-54 代数解交织器模块及其参数设置对话框

在对 Takeshita-Costello 交织信号或 Welch-Costas 交织信号进行解调的过程中，代数解交织器的相应参数应该与产生这个交织信号的代数交织器的参数相同。代数解交织器有以下几

个参数。

■　Type（交织算法）

代数解交织器的交织算法，它可以设置为 Takeshita-Costello 或 Welch-Costas。

■　Number of elements（输入向量的长度）

代数解交织器输入信号向量的长度 $N$。

■　Multiplicative factor（乘法因子）

代数解交织器 Takeshita-Costello 算法的乘法因子 $k$ 。本参数在代数解交织器的交织算法 Type 设置为 Takeshita-Costello 有效。

■　Cyclic shift（循环移位数）

代数解交织器 Takeshita-Costello 算法的循环移位数 $d$。本参数在代数解交织器的交织算法 Type 设置为 Takeshita-Costello 有效，且 $d$ 是介于 1 和 $N-1$ 之间的整数。

■　Primitive element（本原元素）

代数解交织器有限域 GF($N+1$) 的本原元素 $\alpha$。本参数在代数解交织器的交织算法 Type 设置为 Welch-Costas 时有效。

## 7.4.5　随机交织

随机交织是一种随机置换过程。当输入信号向量 $X$ 的长度为 $N$ 时，它随机产生一个长度为 $N$ 的置换向量 $E$，这个置换向量是介于 1 和 $N$ 之间的整数的一个排列。随机交织器根据这个置换向量把输入信号 $X(i)$置换为输出信号 $Y(i)=X(E(i))$，随机解交织器则交织信号实施相反的过程。

### 1．随机交织器

随机交织器（Random Interleaver）对输入信号随机的进行交织。随机交织器的输入信号是一个向量（列向量或行向量），其长度有参数 Number of elements 确定。当输入信号是帧格式数据时，它应该是列向量。随机交织器模块及其参数设置对话框如图 7-55 所示。

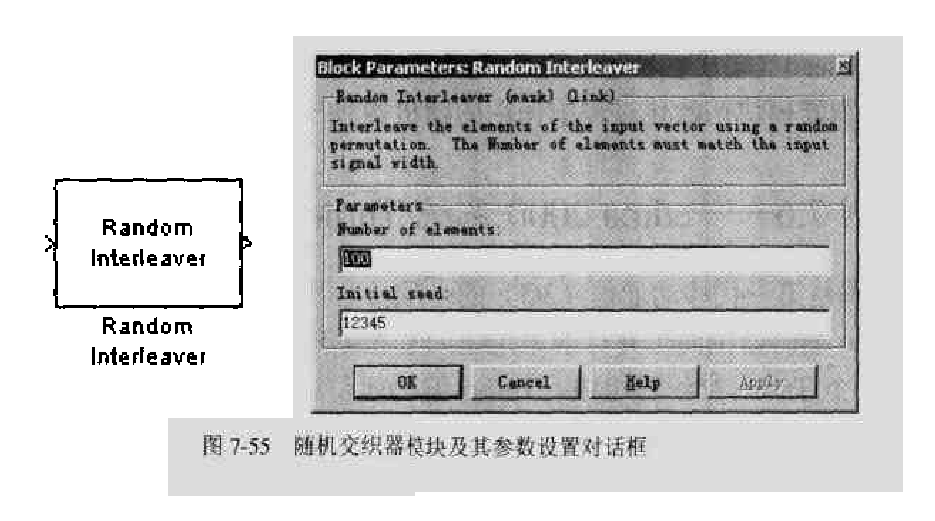



图 7-55　随机交织器模块及其参数设置对话框

对于不同的随机数种子，随机交织器模块产生不同的交织数据；当使用同一个随机数种子时，相同的输入产生相同的输出。如果使用随机数解交织模块对交织数据进行解交织，这两个模块的随机数种子应该相等。

随机交织器模块实际上是由通用块交织器模块构成的，它把通用块交织器模块的置换方式向量设置为一个介于 1 和 $N$ 之间的随机整数向量，然后通过通用块交织器模块完成输入数据得交织过程。随机交织器模块有以下两个参数。

■ Number of elements（输入向量长度）

随机交织器模块输入信号向量的长度。

■ Initial seed（随机数种子）

随机交织器模块初始化过程中使用的随机数种子。

### 2．随机解交织器

随机解交织器（Random Deinterleaver）对随机交织器产生的交织信号进行解交织，还原得到交织前的信号。随机解交织器的输入信号是一个向量（列向量或行向量），其长度有参数 Number of elements 确定。当输入信号是帧格式数据时，它应该是列向量。随机解交织器模块及其参数设置对话框如图 7-56 所示。

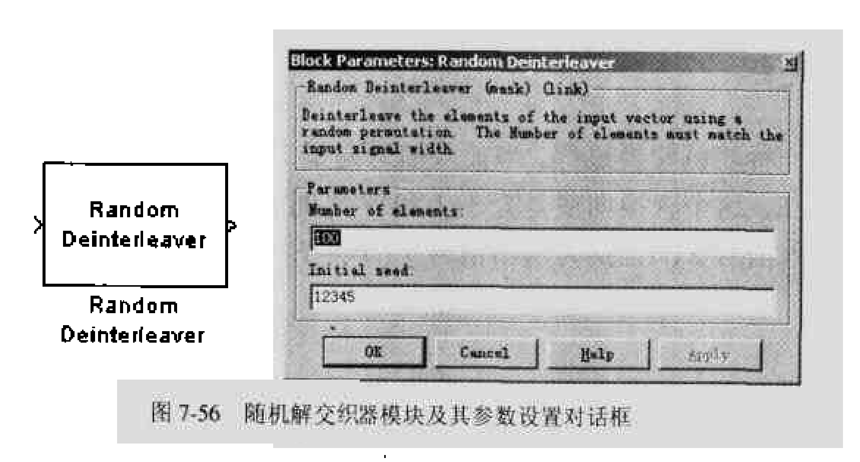



图 7-56 随机解交织器模块及其参数设置对话框

在使用随机数解交织模块对交织数据进行解交织的过程中，这两个模块的随机数种子应该相等。随机解交织器模块有以下两个参数。

■ Number of elements（输入向量长度）

随机解交织器模块输入信号向量的长度。

■ Initial seed（随机数种子）

随机解交织器模块初始化过程中使用的随机数种子。

## 7.4.6 实例 7.6——cdma 2000 系统 Turbo 编码器的实现

Turbo 编码是第三代移动通信（3G）的一项核心技术，目前已经应用于 Wcdma 和 cdma 2000 中。Turbo 编码器由两个并行的系统递归卷积编码器（systematic recursive convolutional encoder）和一个 Turbo 交织器组成，Turbo 编码就是这两个卷积编码器的输出信号经过信号抽取（Puncturing）和重复（Repetition）之后得到的输出信号。从计算机仿真结果看，在交织器长度大于 1000、软判输出卷积解码采用标准的最大后验概率（MAP）算法的条件下，其

性能比约束长度为 9 的卷积码提高 1 至 2.5dB。目前由于交织长度的限制，使得 Turbo 编码无法用于速率较低、时延要求较高的数据（包括语音）传输。同时，基于 MAP 的软输出解码算法所需计算量和存储量较大，而基于软输出 Viterbi 的算法所需迭代次数往往难以保证。本节将以 cdma 2000 协议为基础介绍 Turbo 编码器的工作原理及其实现方法。

1. Turbo 编码器原理

图 7-57 所示是 cdma 2000 系统中的 Turbo 码编码器。在 cdma 2000 系统中，Turbo 编码器有 3 种不同的编码速率，即 1/2、1/3 和 1/4，它们的转换函数可以表示成：

$$G(D)=\left\{1 \quad \frac{n_0(D)}{d(D)} \quad \frac{n_1(D)}{d(D)}\right\},$$

其中反馈函数 $d(D)=1+D^2+D^3$，卷积编码器的两个生成多项式分别为 $n_0(D)=1+D+D^3$，$n_1(D)=1+D+D^2+D^3$。

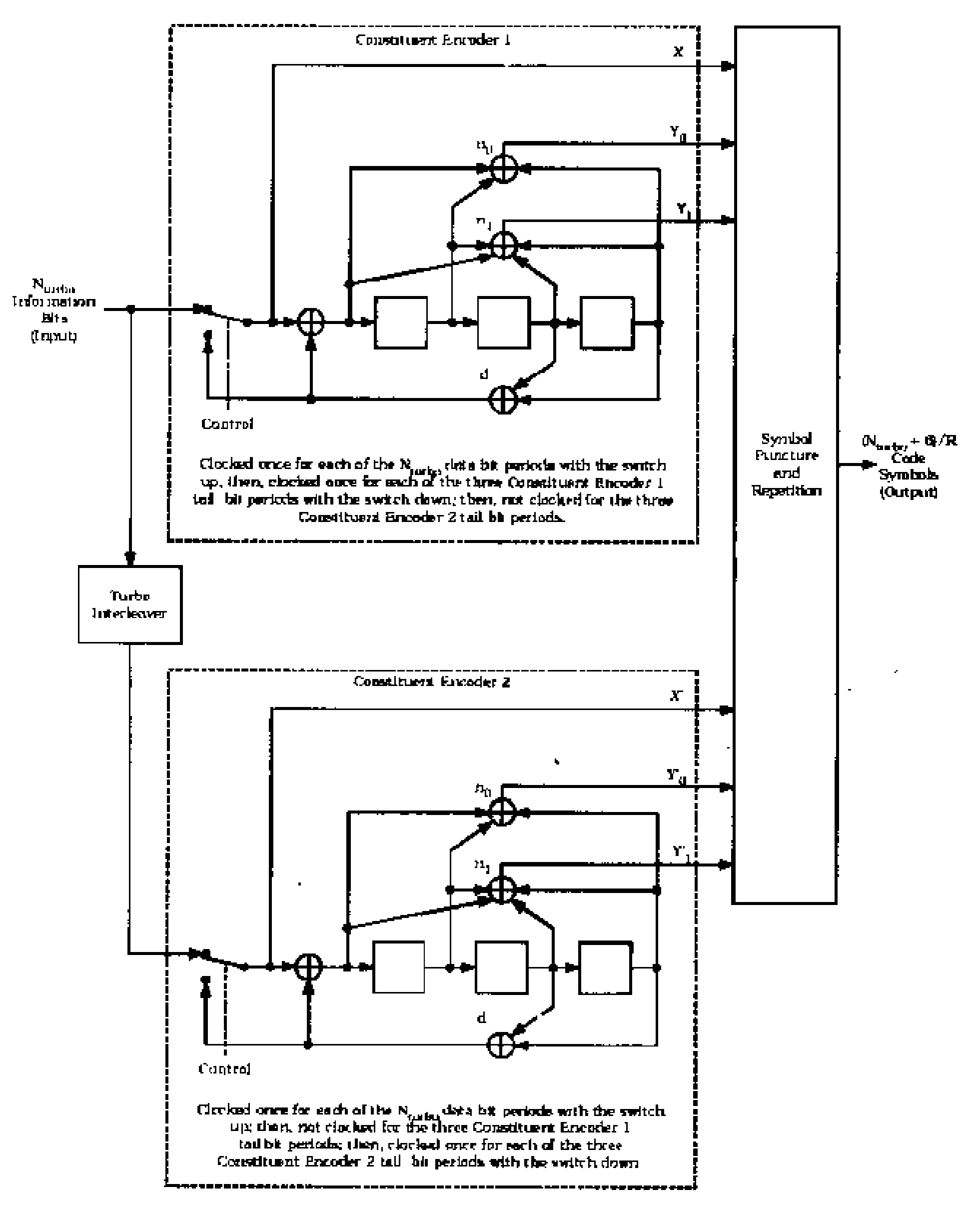



图 7-57　cdma 2000 Turbo 编码器

在 cdma 2000 系统中，Turbo 码编码器的输入信号一方面进入第一个卷积编码（Convolutional Encoder 1）进行卷积编码，另一方面输入信号通过一个 Turbo 交织器（Tu Interleaver），产生的交织信号再通过第二个卷积交织器（Convolutional Encoder 2）。这两个积交织器的约束长度都等于 4，并且分别产生 3 个卷积编码信号（$X$、$Y_0$、$Y_1$ 以及 $X'$、$Y_0'$、$Y_1'$）。这 6 个输出信号并不是都能够成为输出信号，cdma 2000 通过信号抽取和重复技术把其中的某些数据组成 Turbo 编码信号。

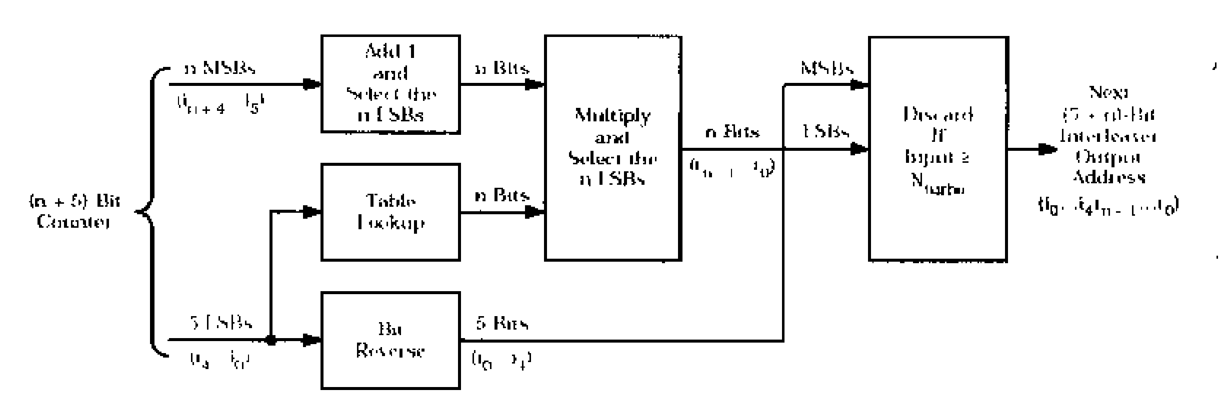



图 7-58　Turbo 交织器的工作流程

Turbo 交织器是一种块交织器。假设 Turbo 交织器输入信号的个数等于 $N_{\text{turbo}}$，每个输入元素依次编号为 $0,1,2,\cdots,N_{\text{turbo}}-1$，则交织器输出信号中第 $i$ 个元素就等于第 $A(i)$ 个输入信号。在计算 $A(i)$ 的过程中，Turbo 交织器采用了一个长度为 $n+5$ bit 的计数器 counter，其中 $n$ 满足条件 $2^{n+5} \geq N_{\text{turbo}}$。计数器 counter 的初始值等于 0，并且在后面的计算过程中依次递增。如果把 counter 写成分成 $i_{n+4}i_{n+3}\cdots i_0$，则可以把它分成两部分，即长度为 $n$ 的最高有效位 MSB（记为 $i_{n+4}\cdots i_5$）和 5 个比特的最低有效位 LSB（记为 $i_4\cdots i_0$）。Turbo 交织器以最低有效位 LSB 为下标查表得到的一个 $n$ bit 的数值，这个数值与最高有效位 MSB 模 $2^n$ 加 1 的结果相乘，乘积的最低 $n$ bit $t_{n-1}\cdots t_0$ 就构成了 $A(i)$ 的低 $n$ 位。同时，计数器 counter 的最低有效位 LSB 在实施反转操作（即把 $i_4\cdots i_0$ 变为 $i_0\cdots i_4$）之后得到的 5 个比特成为 $A(i)$ 的 5 个最高有效位。如果这个过程中得到的 $A(i)$ 大于输入信号的最大序号 $N_{\text{turbo}}-1$，Turbo 交织器将丢弃这个数值，同时将计数器 counter 加 1 之后重新计算 $A(i)$，这个过程将一直持续到 $A(i)$ 满足条件为止。

**表 7-46　　Turbo 交织器查找表**

| 下标 | $n=4$ | $n=5$ | $n=6$ | $n=7$ | $n=8$ | $n=9$ | $n=10$ |
|---|---|---|---|---|---|---|---|
| 0 | 5 | 27 | 3 | 15 | 3 | 13 | 1 |
| 1 | 15 | 3 | 27 | 127 | 1 | 335 | 349 |
| 2 | 5 | 1 | 15 | 89 | 5 | 87 | 303 |
| 3 | 15 | 15 | 13 | 1 | 83 | 15 | 721 |
| 4 | 1 | 13 | 29 | 31 | 19 | 15 | 973 |
| 5 | 9 | 17 | 5 | 15 | 179 | 1 | 703 |
| 6 | 9 | 23 | 1 | 61 | 19 | 333 | 761 |
| 7 | 15 | 13 | 31 | 47 | 99 | 11 | 327 |
| 8 | 13 | 9 | 3 | 127 | 23 | 13 | 453 |
| 9 | 15 | 3 | 9 | 17 | 1 | 1 | 95 |

续表

| 标 | $n=4$ | $n=5$ | $n=6$ | $n=7$ | $n=8$ | $n=9$ | $n=10$ |
|---|---|---|---|---|---|---|---|
| [illegible] | 7 | 15 | 15 | 119 | 3 | 121 | 241 |
| [illegible] | 11 | 3 | 31 | 15 | 13 | 155 | 187 |
| 2 | 15 | 13 | 17 | 57 | 13 | 1 | 497 |
| 3 | 3 | 1 | 5 | 123 | 3 | 175 | 909 |
| 4 | 15 | 13 | 39 | 95 | 17 | 421 | 769 |
| 5 | 5 | 29 | 1 | 5 | 1 | 5 | 349 |
| 6 | 13 | 21 | 19 | 85 | 63 | 509 | 71 |
| 7 | 15 | 19 | 27 | 17 | 131 | 215 | 557 |
| 8 | 9 | 1 | 15 | 55 | 17 | 47 | 197 |
| 9 | 3 | 3 | 13 | 57 | 131 | 425 | 499 |
| 20 | 1 | 29 | 45 | 15 | 211 | 295 | 409 |
| 21 | 3 | 17 | 5 | 41 | 173 | 229 | 259 |
| 22 | 15 | 25 | 33 | 93 | 231 | 427 | 335 |
| 23 | 1 | 29 | 15 | 87 | 171 | 83 | 253 |
| 24 | 13 | 9 | 13 | 63 | 23 | 409 | 677 |
| 25 | 1 | 13 | 9 | 15 | 147 | 387 | 717 |
| 26 | 9 | 23 | 15 | 13 | 243 | 193 | 313 |
| 27 | 15 | 13 | 31 | 15 | 213 | 57 | 757 |
| 28 | 11 | 13 | 17 | 81 | 189 | 501 | 189 |
| 29 | 3 | 1 | 5 | 57 | 51 | 313 | 15 |
| 30 | 15 | 13 | 15 | 31 | 15 | 489 | 75 |
| 31 | 5 | 13 | 33 | 69 | 67 | 391 | 163 |

表 7-46 是 cdma 2000 协议规范中当 $n$ 介于 4～10 之间时 Turbo 交织器的查找表，其中下标表示计数器 counter 的 5bit 最低有效位 LSB。在计算过程中，Turbo 交织器以 LSB 为下标就可以从表中得到一个长度为 $n$bit 的数据。

在任意一个时刻，两个卷积编码器分别输出 3bit 的编码符号，这些符号按顺序组合成 $XY_0Y_1X'Y_0'Y_1'$，然后由 Turbo 编码器的抽取和重复模块（Puncture and Repetition）抽取其中的某些信号作为 Turbo 编码输出。抽取和重复操作是以两个符号周期为单位进行的，即对于卷积编码符号 $XY_0Y_1X'Y_0'Y_1'XY_0Y_1X'Y_0'Y_1'$，码率为 1/2 的 Turbo 编码器产生的输出信号为 $XY_0XY_0'$，码率为 1/3 的 Turbo 编码器产生的输出信号为 $XY_0Y_0'XY_0Y_0'$，而码率为 1/4 的 Turbo 编码器产生的输出信号为 $XY_0Y_1Y_1'XY_0Y_0'Y_1'$。

当所有的输入信号都进入 Turbo 编码器完成交织和编码之后，两个卷积编码器还需要运行 6 个符号周期，产生 36 个输出符号。这些输出符号仍然要执行抽取和重复操作。对于前 3 个周期的输出符号 $XY_0Y_1X'Y_0'Y_1'$，1/2 码率的 Turbo 编码器输出 $XY_0$，1/3 码率的 Turbo 编码器输出 $XXY_0$，1/4 码率的 Turbo 编码器则输出 $XXY_0Y_1$。对于后 3 个周期的输出符号 $XY_0Y_1X'Y_0'Y_1'$，1/2 码率的 Turbo 编码器输出 $X'Y_0'$，1/3 码率的 Turbo 编码器输出 $X'X'Y_0'$，1/4 码率的 Turbo 编码器则输出 $X'X'Y_0'Y_1'$。这样，对于长度为 $N_{turbo}$ 的输入信号，码率为

$1/n, n=2,3,4$ 的 Turbo 编码器产生的编码符号数等于$(N_{turbo}+6)\cdot n$。

## 2. Turbo 编码器的实现

前面我们介绍了 cdma 2000 系统 Turbo 编码器的工作原理，本节我们将通过 Simulink 实现这个 Turbo 编码器。图 7-59 所示是我们这个仿真模型的结构图，其中 Bernoulli Binary Generator（贝努利二进制序列生成器模块）产生一个长度为 378bit 的数据帧，这个数据帧通过 Turbo Encoder（Turbo 编码器）进行 Turbo 编码，并且把 Turbo 编码信号通过 To Workspace（工作区写入模块）保存到工作区变量 Signal 中。

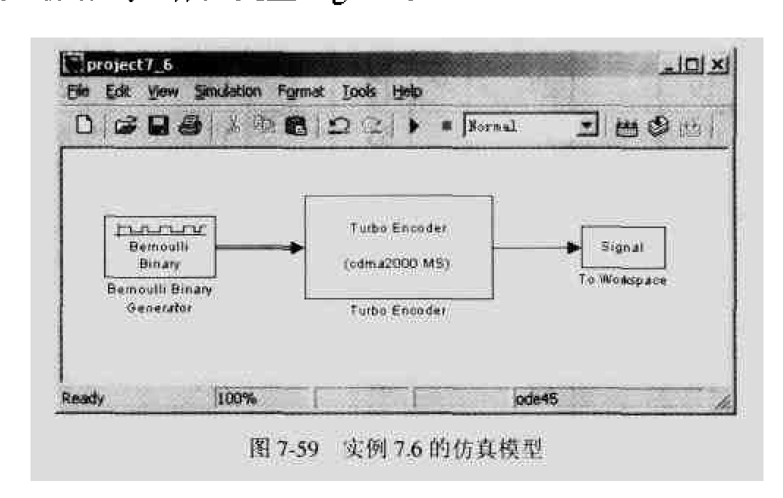



图 7-59 实例 7.6 的仿真模型

表 7-47 和表 7-48 列出了贝努利二进制序列产生器模块和工作区写入模块的参数设置情况，其中贝努利二进制序列产生器模块以 1 秒为周期产生一个长度为 378bit 的数据帧。根据 cdma 2000 协议规范，这个数据帧的长度可以是 378bit、570bit、762bit、1,146bit、1 530bit、2 298bit、3 066bit、4 602bit、6 138bit、9 210bit、12 282bit 以及 20 730bit。

**表 7-47 Bernoulli Binary Generator（贝努利二进制序列产生器模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | 61 |
| Sample time | 1/378 |
| Frame-based outputs | Checked |
| Samples per frame | 378 |

**表 7-48 To Workspace（工作区写入模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | To Workspace |
| Variable name | Signal |
| Limit data points to last | inf |
| Decimation | 1 |
| Sample time (-1 for inherited) | -1 |
| Save format | Array |

本节的任务是通过 Simulink 设计一个 Turbo 编码器。图 7-60 所示是这个模块的参数设置对话框，它有两个参数： Input Frame Length（输入信号的数据帧长度）以及 Code Rate（码率），其中 Code Rate 参数可以选择 1/2、1/3 或 1/4。

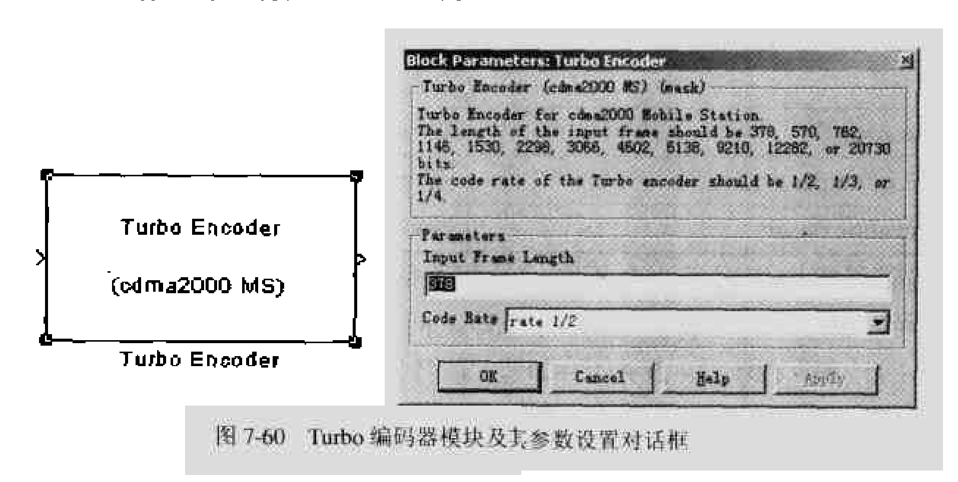



图 7-60　Turbo 编码器模块及其参数设置对话框

图 7-61 所示是 Turbo 编码器模块的内部结构，它基本上是按照 cdma 2000 协议规范进行设计的。从图 7-61 所示中可以看到，Turbo 编码器模块由两个卷积编码器（Convolutional Encoder 1 和 Convolutional Encoder 2）、一个 Turbo 交织器（Turbo Interleaver）和一个 Turbo 抽取模块（Turbo Puncture）组成。

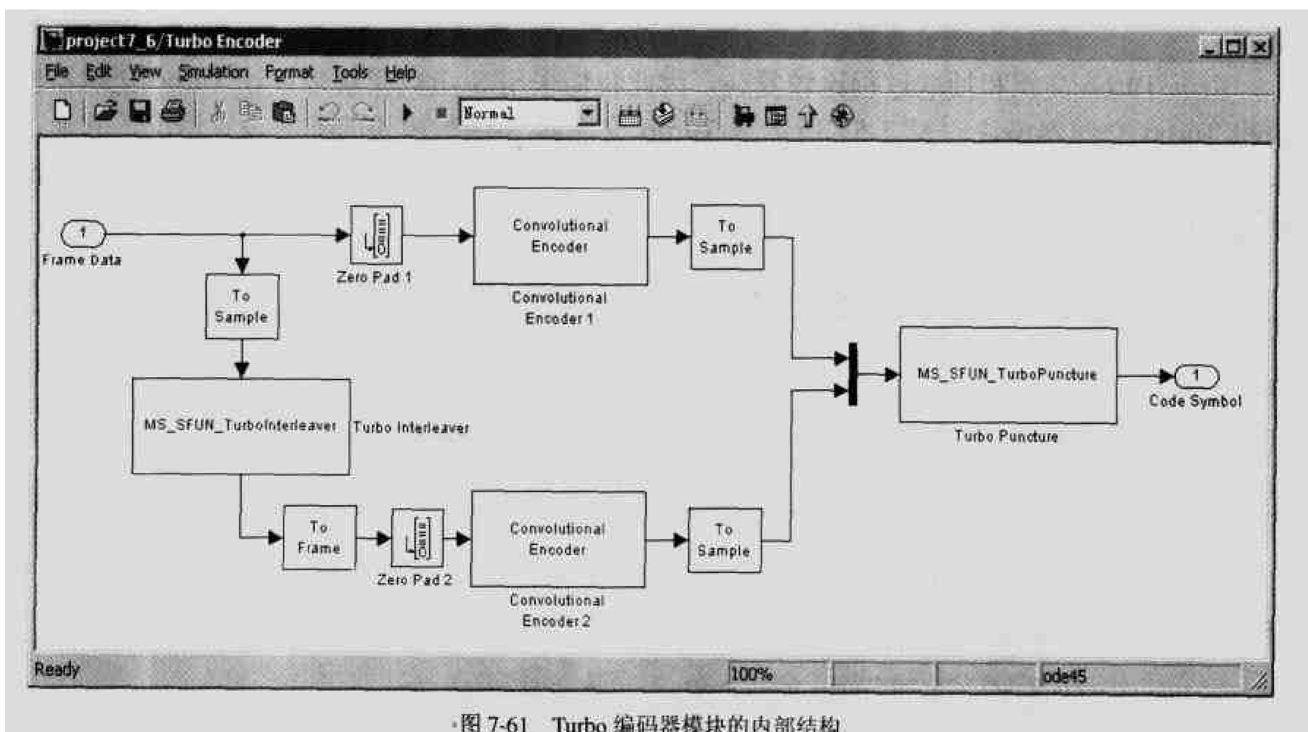



图 7-61　Turbo 编码器模块的内部结构

Turbo 编码器的输入信号在进入卷积编码器之前首先通过零填充模块（Zero Pad 1 和 Zero Pad 2），在输入数据帧中添加 6 个 0，使得卷积编码器在完成输入数据帧的编码过程之后还能够再工作 6 个符号周期。表 7-49 是这两个零填充模块的参数设置情况。

表 7-49　　**Zero Pad（零填充模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Zero Pad |
| Pad signal at | End |
| Pad along | Columns |
| Number of output rows | User-specified |
| Specified number of output rows | xFrameLength+6 |
| Action when truncation occurs | None |

Turbo 编码器模块使用了两个相同的卷积编码器。这个卷积编码器带有反馈，其反馈多项式为 $d(D)=1+D^2+D^3$（即八进制的 13）。同时，卷积编码器的两个生成多项式分别为 $n_0(D)=1+D+D^3$ 和 $n_1(D)=1+D+D^2+D^3$，它们分别对应于八进制数 15 和 17。因此，在 MATLAB 中卷积编码器可以表示为 poly2trellis(4, [13 15 17], 13)。表 7-50 所示是这两个卷积编码器模块的参数设置。

表 7-50　　**Convolutional Encoder（卷积编码器模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Convolutional Encoder |
| Trellis structure | poly2trellis(4, [13 15 17], 13) |
| Reset | None |

由于 Turbo 交织和抽取过程比较复杂，因此我们采用 S-函数模块来设计 Turbo 交织器模块和 Turbo 交织器模块。表 7-51 所示是 Turbo 交织器 S-函数模块的参数设置情况，它通过 S-函数 MS_SFUN_TurboInterleaver 实现计算功能，并且带有一个表示输入数据帧长度的参数 xFrameLength。

表 7-51　　**S-Function（Turbo 交织器 S-函数模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | S-Function |
| S-function name | MS_SFUN_TurboInterleaver |
| S-function parameters | xFrameLength |

下面列出了 S-函数 MS_SFUN_TurboInterleaver 的代码，它采用 M 文件 S-函数的格式，并且带有一个参数 interleaver_size。

```
function [sys,x0,str,ts] = MS_SFUN_TurboInterleaver(t,x,u,flag, interleaver_size)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% M 文件 S-函数的主体部分

% 函数名称：MS_SFUN_TurboInterleaver

% 主要功能：根据输入参数 flag 的数值调用相应的函数

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
switch flag,
  case 0,
    % 当 flag 等于 0 时调用 mdlInitializeSizes 函数执行初始化
    [sys,x0,str,ts]=mdlInitializeSizes(interleaver_size);
  case 2,
    % 当 flag 等于 2 时调用 mdlUpdate 函数计算离散状态的数值
    sys=mdlUpdate(t,x,u,interleaver_size);
  case 3,
    % 当 flag 等于 3 时调用 mdlOutputs 函数计算输出信号的数值
    sys=mdlOutputs(t,x,u,interleaver_size);
  case {1, 4, 9}
    % 当 flag 等于 1、4 或 9 时没有相应的操作（没有连续状态）
    sys=[];
  otherwise
    % 当 flag 等于其他数值时表示仿真过程出错
    error(['Unhandled flag = ',num2str(flag)]);
end
% end MS_SFUN_TurboInterleaver
%
%=============================================================================
% mdlInitializeSizes
% S-函数的初始化
% 向 Simulink 返回 S-函数各种信号的长度、初始设置和抽样时间设置
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes(interleaver_size)
% 调用 simsizes 获得一个用于存放长度信息的结构
sizes = simsizes;
% 设置连续状态的个数
sizes.NumContStates = 0;
```

```
% 设置离散状态的个数
% x(1)表示交织器参数 n
sizes.NumDiscStates = 1;
% 设置输出信号的个数为动态确定
sizes.NumOutputs = -1;
% 设置输入信号的个数为动态确定
sizes.NumInputs = -1;
% 设置直接反馈的状态：
%    0 表示没有直接反馈
%    1 表示存在直接反馈
sizes.DirFeedthrough = 1;
% 设置抽样时间的个数（大于等于 1）
sizes.NumSampleTimes = 1;
% 通过 simsizes 把 sizes 结构返回给 sys
sys = simsizes(sizes);
%
% 设置 S-函数的初始状态 x0
n = mdlInterleaverInit(interleaver_size);
x0 = n;
%
% 设置 S-函数的保留参数 str（应该设置为空向量[ ]）
str = [];
%
% 初始化抽样时间
ts   = [-1 0];
% end mdlInitializeSizes
%
%=============================================================================
% mdlUpdate
```

```
% 更新 S-函数的离散状态并且向 Simulink 返回这些状态的数值
%==============================================================================
function sys=mdlUpdate(t,x,u,interleaver_size)
% 确定 Turbo 编码器的参数 n
n = mdlInterleaverInit(interleaver_size);
% 更新离散状态
sys = n;
% end mdlUpdate
%
%==============================================================================
% mdlOutputs
% 计算 S-函数的输出信号并且返回给 Simulink 作为模块的输出
%==============================================================================
function sys=mdlOutputs(t,x,u,interleaver_size)
% 离散状态 x(1)是 Turbo 编码器的参数 n
n = x(1);
% A 表示 Turbo 交织器的输入信号
A = zeros(1,interleaver_size);
% counter 是计数器，初始值等于 0
counter = 0;
for index = 0:interleaver_size-1
    % label 是一个信号量，用于表示是否应该重新计算 Turbo 交织的下标
    label = 1;
    while label==1
        % 从 counter 中获取 n 个最高有效位 msb
        msb = bitshift(bitand(counter,(2^n-1)*32),-5);
        % 从 counter 中获取 5 个最低有效位 lsb
        lsb = bitand(counter,31);
        % a 是最高有效位模 2^n 加 1 的结果
```

```
            a = mod(msb + 1, 2^n);
            % b 表示最低有效位的位反转（即把 i4 i3 i2 i1 i0 转换成 i0 i1 i2 i3 i4）
            b = 0;
            for k = 1:5
                if bitget(lsb, k) == 1
                    b = bitset(b, 6-k);
                end
            end
            % 根据最低有效位 lsb 和 Turbo 编码参数 n 查表得到 c
            c = mdlLookupTable(lsb, n);
            % d 是 a 和 c 的乘积模 2^n 之后得到的结果
            d = mod(a*c, 2^n);
            % b 和 d 分别是 e 的最高有效位和最低有效位
            e = b*2^n + d;
            % 计数器 counter 的数值增加 1
            counter = mod(counter + 1, 2^(n+5));
            % 检查 e 的数值是否合法
            if e < interleaver_size
                % 如果 e 小于交织器输入信号的最大下标则设置信号量 label 为 0
                % 否则，信号量 label 仍然等于 1，重新在这个循环过程中计算 e
                label = 0;
            end
        end
        % A 的第 index+1 个元素等于输入信号的第 e+1 个元素
        A(index+1) = u(e+1);
end
% 输出信号等于 A
sys = A;
% end mdlOutputs
```

```
%
%=========================================================================
% mdlInterleaverInit
% 初始化交织器的长度
%=========================================================================
function n=mdlInterleaverInit(interleaver_size)
% 根据交织器长度确定 Turbo 编码的参数 n
switch interleaver_size
    % 合法的 Turbo 交织器长度是 378, 570, 762, 1146, 1530, 2298, 3066, 4602, 6138, 9210, 12282 和 20730
    case {378, 570, 762, 1146, 1530, 2298, 3066, 4602, 6138, 9210, 12282, 20730}
        % 计算 Turbo 编码的参数 n，使之满足条件 2^(n+5)>=interleaver_size
        n = ceil(log2(interleaver_size/32));
    % interleaver_size 的其他数值都是不合法的
    otherwise
        error('Error: Invalid Interleaver Size in MS_SFUN_TurboInterleaver');
end
%end mdlInterleaverInit
%
%=========================================================================
% mdlLookupTable
% 根据 Turbo 交织器的下标查找对应的数值
%=========================================================================
function index=mdlLookupTable(lsb, n)
% 查找表是一个 32 行 7 列的矩阵
% 矩阵第 i 行的元素表示与下标 lsb 等于 i-1 时的数值
% 矩阵第 j 列的元素表示当 Turbo 交织器参数 n 等于 j+3 时的数值
LookupTable = [5 27 3 15 3 13 1; 15 3 27 127 1 335 349;5 1 15 89 5 87 303;15 15 13 1 83 15 721;
    1 13 29 31 19 15 973;9 17 5 15 179 1 703;9 23 1 61 19 333 761;15 13 31 47 99 11 327;
    13 9 3 127 23 13 453;15 3 9 17 1 1 95;7 15 15 119 3 121 241;11 3 31 15 13 155 187;
```

```
    15 13 17 57 13 1 497;3 1 5 123 3 175 909;15 13 39 95 17 421 769;5 29 1 5 1 5 349;
    13 21 19 85 63 509 71;15 19 27 17 131 215 557;9 1 15 55 17 47 197;3 3 13 57 131 425 499;
    1 29 45 15 211 295 409;3 17 5 41 173 229 259;15 25 33 93 231 427 335;1 29 15 87 171 83 253;
    13 9 13 63 23 409 677;1 13 9 15 147 387 717;9 23 15 13 243 193 313;15 13 31 15 213 57 757;
    11 13 17 81 189 501 189;3 1 5 57 51 313 15;15 13 15 31 15 489 75;5 13 33 69 67 391 163];
% 根据 lsb 和 n 的数值得到 LookupTable 的数值
index = LookupTable(lsb+1, n-3);
%end mdlLookupTable
```

Turbo 抽取器通过 S-函数 MS_SFUN_TurboPuncture 实现数据的抽取和重复过程。表 7-52 所示是 Turbo 抽取器 S-函数模块的参数设置情况，它带有两个参数，即数据帧长度 xFrameLength 和编码速率 xCodeRate。

**表 7-52　S-Function（Turbo 抽取器 S-函数模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | S-Function |
| S-function name | MS_SFUN_TurboPuncture |
| S-function parameters | xFrameLength,xCodeRate |

M 文件 S-函数 MS_SFUN_TurboPuncture 的代码如下：

```
function [sys,x0,str,ts] = MS_SFUN_TurboPuncture(t,x,u,flag, interleaver_size, code_rate)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% M 文件 S-函数的主体部分
% 函数名称：MS_SFUN_TurboPuncture
% 主要功能：根据输入参数 flag 的数值调用相应的函数
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch flag,
  case 0,
    % 当 flag 等于 0 时调用 mdlInitializeSizes 函数执行初始化
    [sys,x0,str,ts]=mdlInitializeSizes(interleaver_size, code_rate);
  case 3,
    % 当 flag 等于 3 时调用 mdlOutputs 函数计算输出信号的数值
    sys=mdlOutputs(t,x,u,interleaver_size, code_rate);
  case {1, 2, 4, 9}
    % 当 flag 等于 1、2、4 或 9 时没有相应的操作
    sys=[];
  otherwise
```

```
    % 当 flag 等于其他数值时表示仿真过程出错
    error(['Unhandled flag = ',num2str(flag)]);
end
% end MS_SFUN_TurboPuncture
%
%=============================================================================
% mdlInitializeSizes
% S-函数的初始化
% 向 Simulink 返回 S-函数各种信号的长度、初始设置和抽样时间设置
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes(interleaver_size, code_rate)
% 调用 simsizes 获得一个用于存放长度信息的结构
sizes = simsizes;
% 设置连续状态的个数
sizes.NumContStates = 0;
% 设置离散状态的个数
sizes.NumDiscStates = 0;
% len 表示输入信号的长度
len = (interleaver_size + 6) * 6;
% 根据 Turbo 编码的码率 code_rate 确定输出信号的长度
switch code_rate
    case 1
        % 1/2 码率输出信号长度等于输入信号长度的 1/3
        out_len = len/3;
    case 2
        % 1/3 码率输出信号长度等于输入信号长度的 1/2
        out_len = len/2;
    case 3
        % 1/4 码率输出信号长度等于输入信号长度的 2/3
        out_len = len * 2/3;
    otherwise
        error('Error: Invalid Code Rate in MS_SFUN_TurboPuncture');
end
% 设置输出信号的长度
sizes.NumOutputs = out_len;
% 设置输入信号的长度
```

```
sizes.NumInputs = len;
% 设置直接反馈的状态:
%    0 表示没有直接反馈
%    1 表示存在直接反馈
sizes.DirFeedthrough = 1;
% 设置抽样时间的个数（大于等于 1）
sizes.NumSampleTimes = 1;
% 通过 simsizes 把 sizes 结构返回给 sys
sys = simsizes(sizes);
%
% 设置 S-函数的初始状态 x0（没有离散状态）
x0 = [];
%
% 设置 S-函数的保留参数 str（应该设置为空向量[]）
str = [];
%
% 初始化抽样时间
ts   = [-1 0];
% end mdlInitializeSizes
%
%==============================================================================
% mdlOutputs
% 计算 S-函数的输出信号并且返回给 Simulink 作为模块的输出
%==============================================================================
function sys=mdlOutputs(t,x,u,interleaver_size, code_rate)
% 计算输入信号的长度
len = (interleaver_size + 6) * 6;
% 输入信号向量的前半部分表示第一个卷积编码器的编码信号，其下标由 index1 表示
index1 = 1;
% 输入信号向量的后半部分表示第二个卷积编码器的编码信号，其下标由 index2 表示
index2 = len/2 + 1;
% A 用于保存调换顺序后的输入信号
A = zeros(1,len);
% 把输入信号调换顺序，使得同一时刻两个编码器产生的编码信号在 A 中连续存放
for i = 1:len
    % A 中元素以 6 个为一组
```

```
    if mod(i-1,6) < 3
        % 前三个元素表示第一个卷积编码器的编码信号，它在输入信号 u 中的下标由 index1 确定
        A(i) = u(index1);
        % 增加计数器 index1 的数值
        index1 = index1 + 1;
    else
        % 后三个元素表示第二个卷积编码器的编码信号，它在输入信号 u 中的下标由 index2 确定
        A(i) = u(index2);
        % 增加计数器 index2 的数值
        index2 = index2 + 1;
    end
end
% 根据编码速率确定信号抽取和重复的方式以及输出信号的长度
% 1 表示 1/2 码率，2 表示 1/3 码率，3 表示 1/4 码率
% 抽取方式向量中 0 表示删除该信号，1 表示输出该信号，2 表示重复输出该信号
switch code_rate
    case 1
        % 1/2 码率
        % pattern 表示对 A 中前 len-36 个元素进行抽取的方式
        pattern = [1 1 0 0 0 0 1 0 0 0 1 0];
        % pattern1 表示对 A 中从 len-35 个元素开始的 18 个信号的抽取方式
        pattern1 = [1 1 0 0 0 0];
        % pattern2 表示对 A 中最后 18 个元素的抽取方式
        pattern2 = [0 0 0 1 1 0];
        % out_len 表示输出信号的长度
        out_len = len/3;
    case 2
        % 1/3 码率
        pattern = [1 1 0 0 1 0 1 1 0 0 1 0];
        pattern1 = [2 1 0 0 0 0];
        pattern2 = [0 0 0 2 1 0];
        out_len = len/2;
    case 3
        % 1/4 码率
        pattern = [1 1 1 0 0 1 1 1 0 0 1 1];
        pattern1 = [2 1 1 0 0 0];
```

```
            pattern2 = [0 0 0 2 1 1];
            out_len = len * 2/3;
    otherwise
            error('Error: Invalid Code Rate in MS_SFUN_TurboPuncture');
end
% B 表示输出信号
B = zeros(1,out_len);
% index 表示输出信号的下标
index = 1;
% 根据抽取方式 pattern 对 A 中前 len-36 个元素进行抽取
for i = 1:(len-36)
    % 如果 pattern 等于 1 则输出 A 中的元素 A(i)
    % 否则忽略 A(i)
    if pattern(mod(i-1,12)+1) == 1
        B(index) = A(i);
        index = index + 1;
    end
end
% 根据抽取方式 pattern1 对从 len-35 个元素开始的 18 个信号进行抽取和重复
for i = (len-36)+1:len-18
    % 如果 pattern 等于 1 则输出 A 中的元素 A(i)一次
    % 如果 pattern 等于 2 则输出 A 中的元素 A(i)两次
    % 否则忽略 A(i)
    for k = 1:pattern1(mod(i-1,6)+1)
        B(index) = A(i);
        index = index + 1;
    end
end
% 根据抽取方式 pattern2 对最后 18 个信号进行抽取和重复
for i = (len-18)+1:len
    % 如果 pattern 等于 1 则输出 A 中的元素 A(i)一次
    % 如果 pattern 等于 2 则输出 A 中的元素 A(i)两次
    % 否则忽略 A(i)
    for k = 1:pattern2(mod(i-1,6)+1)
        B(index) = A(i);
        index = index + 1;
```

```
    end
end
% 输出信号等于 B
sys = B;
% end mdlOutputs
```

由于 M 文件 S-函数不能接受帧格式的输入数据，因此在使用 Turbo 交织模块和 Turbo 抽取模块之前，必须通过帧状态转换模块（To Sample）把帧格式的输入数据转换成抽样格式。同时，卷积编码器模块只能够接收帧格式的输入数据，因此 Turbo 交织模块的输出信号还需要通过另外一个帧状态转换模块（To Frame）还原为帧格式数据。

另外，复用器模块（Mux）把两个卷积交织器的输出信号组合成一个向量，然后把这个向量作为 Turbo 抽取模块的输入信号。Simulink 的复用器模块把第二个卷积编码信号附加在第一个卷积编码信号的后面，因此在 M 文件 S-函数 MS_SFUN_TurboPuncture 中，为了处理的方便，还需要把这个向量的数据顺序调换一下，使得每个时刻产生的 6 个卷积编码符号在向量中连续放置。

最后我们把 Turbo 编码器模块转换成一个封装子系统，使之具有与 Simulink 模块相同的外部特征。在 Simulink 中选中这个子系统，单击鼠标右键，从弹出式菜单中选择 Mask subsystem 后就创建了一个封装子系统。这时候我们可以选择 Edit mask，在 Mask Editor 窗口中设置封装模块的外观和参数。Mask Editor 窗口由 Icon、Parameters、Initialization 和 Documentation 4 个面板组成，我们把 Icon 面板的 Drawing commands 设置为 disp('Interleaver\n\n(cdma 2000 MS)')，并且在 Parameters 面板中创建两个封装子系统内部变量 xFrameLength 和 xCodeRate，它们分别表示输入信号的数据帧长度以及 Turbo 编码器的编码速率。其中，xCodeRate 是一个 popup 类型的变量，它只允许用户从 3 个选择项 rate 1/2、rate 1/3 和 rate 1/4 中选择一个项目，然后向 Simulink 返回一个数值（1、2 或 3）表示用户的当前选择。

我们可以通过 Simulink 启动仿真程序运行。仿真结束之后 Turbo 编码信号保存在工作区变量 Signal 中。这个编码信号是一个二进制序列，我们只能通过输出信号的长度来验证 Turbo 编码器的正确性，除此之外这个输出信号没有什么意义。在本书的第 10 章中，我们将介绍 cdma 2000 系统仿真模型的设计和实现，因此本节介绍的这个 Turbo 编码器可以应用到这个系统仿真模型中，在整个仿真系统中研究和验证 Turbo 编码的性能。

## 7.5 卷积交织

卷积交织是与块交织不同的一种交织方法。在块交织中，一段时间内的输出信号只与该段时间内的输入信号有关；而在卷积交织中，某段时间内的输出信号不仅包含了这段时间内的输入信号，还可能有该段时间之前的输入信号。卷积交织广泛应用于现代移动通信系统中，尤其是与卷积编码同时使用时能够获得更佳理想的效果。在 MATLAB 中，卷积交织有 3 种模块，即复用交织器、卷积交织器以及螺旋交织器。本节将依次介绍这 3 种交织器及其相应的解交织器的原理和使用方法。

## 7.5.1 复用交织

复用交织器是一种基本的卷积交织模块，它采用一系列并行的移位寄存器来实现对输入信号的交织，这些并行的移位寄存器可以具有不同数目的寄存器。通过指定不同的并行移位寄存器的数目可以实现不同的复用交织方案，它们具有不同的解交织时延。本节介绍复用交织器和复用解交织器。

### 1. 复用交织器

复用交织器（General Multiplexed Interleaver）采用多个并行的移位寄存器对输入信号实施交织，这些并行的移位寄存器可以设置不同数目的寄存器，从而产生不同的输出时延。复用交织器模块的输入信号可以是标量，也可以是帧格式的列向量，其中每个元素可以是实信号，也可以是复信号。复用交织器模块的输出信号的抽样间隔与输入信号相同。复用交织器模块及其参数设置对话框如图 7-62 所示。

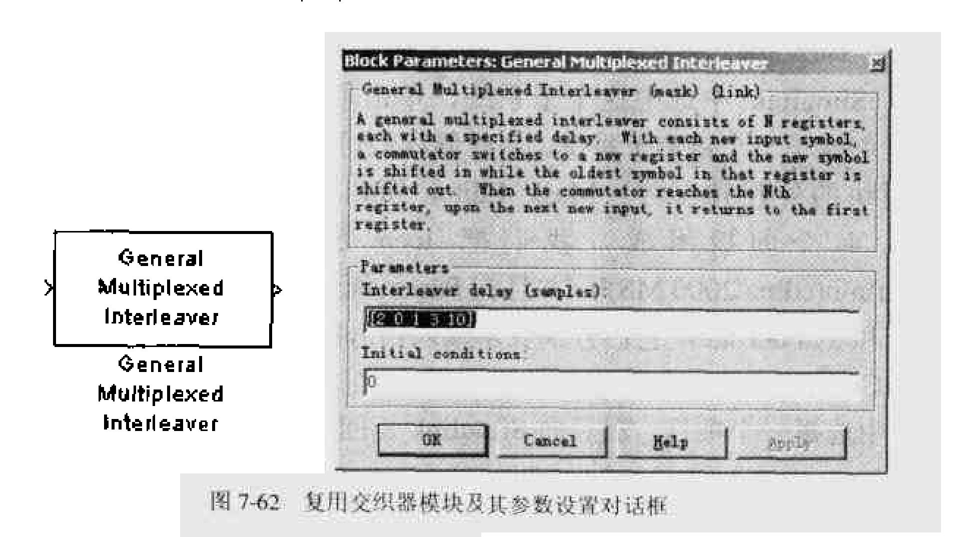



图 7-62 复用交织器模块及其参数设置对话框

复用交织器可以看作是 $N$ 个并行的移位寄存器，其中第 $i(1\leq i\leq N)$行移位寄存器有 $L_i(1\leq i\leq N)$个寄存器。输入信号依次进入这 $N$ 个移位寄存器，同时输出这一行的一个信号。例如，当 $N$ 等于 6，且这六行移位寄存器分别具有 2、0、1、3、1、0 个寄存器时，复用交织器的结构如图 7-63 所示。

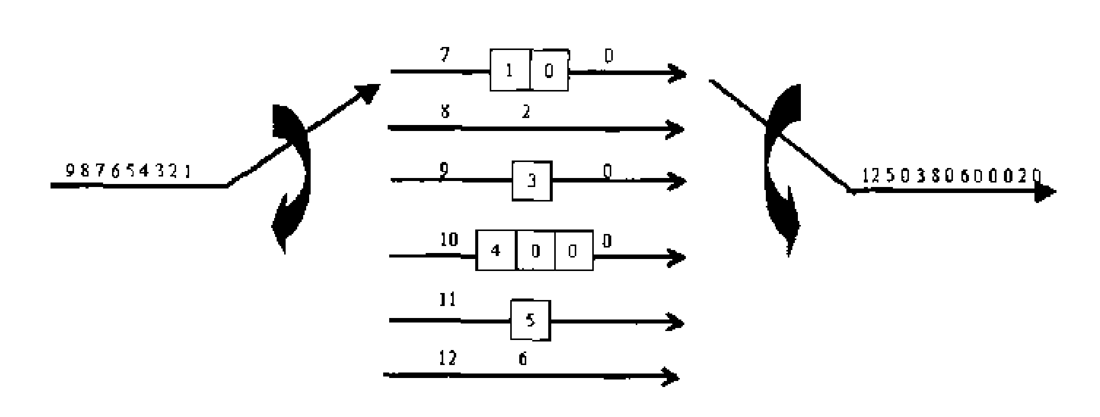



图 7-63 一个复用交织器的结构示例

在图 7-63 所示中的复用交织器中，第一行有两个寄存器，在初始状态，这两个寄存器的数值都等于 0（通过参数 Initial conditions 可以设置初始值）。假设输入信号依次为 1,2,3,…，在第一个抽样周期内，输入信号 1 进入第一行最左边的移位寄存器，同时输出这一行最右边的寄存器的数值 0，这时候移位寄存器状态变成 10。在第二个抽样周期，复用交织器的两个开关同时切换到第二行移位寄存器，由于这一行没有设置寄存器，因此输入信号 2 直接进入输出数据流。同理，复用交织器的开关依次切换到第三、四、五、六行移位寄存器，在第六个抽样周期，复用交织器已经产生的输出数据是 0,2,0,0,0,6。在第七个抽样周期到来的时候，复用交织器的开关重新切换到第一行移位寄存器，输入数据 7 保存到最左边的寄存器中，这个寄存器中原有的数据 1 向右移动到右边的寄存器，同时输出最右边寄存器中的数据 0。复用交织器按照上面的过程循环工作，从而实现对输入数据的交织。复用交织器模块有以下两个参数。

■　Interleaver delay (samples)（交织延时）

复用交织器的交织时延列向量 $L=[L_1,L_2,\cdots L_N]'$，向量的长度等于复用交织器中移位寄存器的行数 $N$，向量中的每个元素表示该行移位寄存器中寄存器的数目。当某一行移位寄存器的数目等于 0 时表示没有时延。当输入信号是抽样信号时，本参数也可以是一个行向量。例如，图 7-63 所示的复用交织器的交织时延等于[2,0,1,3,1,0]′。

■　Initial conditions（初始状态）

复用交织器模块中的移位寄存器的初始状态。当本参数是一个标量时，复用交织器中所有寄存器的初始状态都等于这个数值；当本参数设置位一个向量时，向量的长度与交织时延向量的长度相同，其中每一个元素对应于这一行的移位寄存器的初始状态（同一行移位寄存器中的各个寄存器具有相同的初始状态）。当交织时延向量中的某个元素等于 0 时，初始状态向量中与之对应的元素的数值不起作用。

### 2. 复用解交织器

复用解交织器（General Multiplexed Deinterleaver）对复用交织器产生的交织信号实施解交织，得到交织前的信号。复用解交织器模块的参数应该与相应的复用交织器模块的参数相同，它的输入信号可以是标量，也可以是帧格式的列向量，同时，输出信号的抽样间隔与输入信号相同。复用解交织器模块及其参数设置对话框如图 7-64 所示。

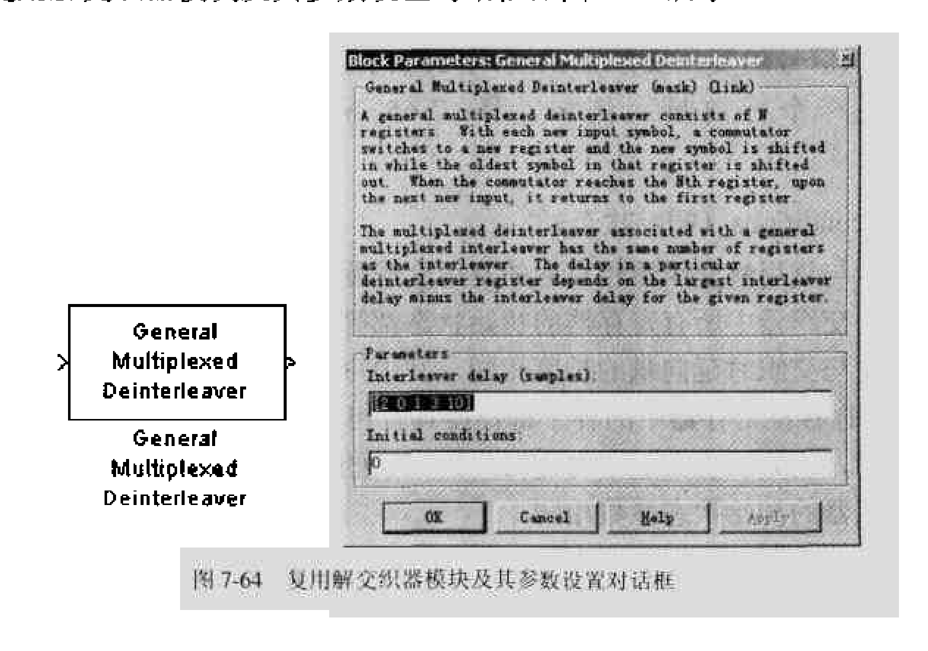


图 7-64　复用解交织器模块及其参数设置对话框

与复用交织器类似地，复用解交织器也采用 $N$ 个并行的移位寄存器对信号进行解交织。假设产生交织信号的复用交织器第 $i(1\leqslant i\leqslant N)$行移位寄存器有 $L_i$ 个寄存器，则在复用解交织器中，第 $i$ 行移位寄存器有 $\max\limits_{1\leqslant k\leqslant N}(L_k)-L_i$ 个寄存器。图 7-65 是与图 7-63 所示的复用交织器相对应的复用解交织器的结构图，其中 $N$ 等于 6，复用交织器 6 个移位寄存器分别具有 2、0、1、3、1、0 个寄存器，因此，在复用解交织器中，各行移位寄存器的寄存器数目分别为 1、3、2、0、2 和 3。

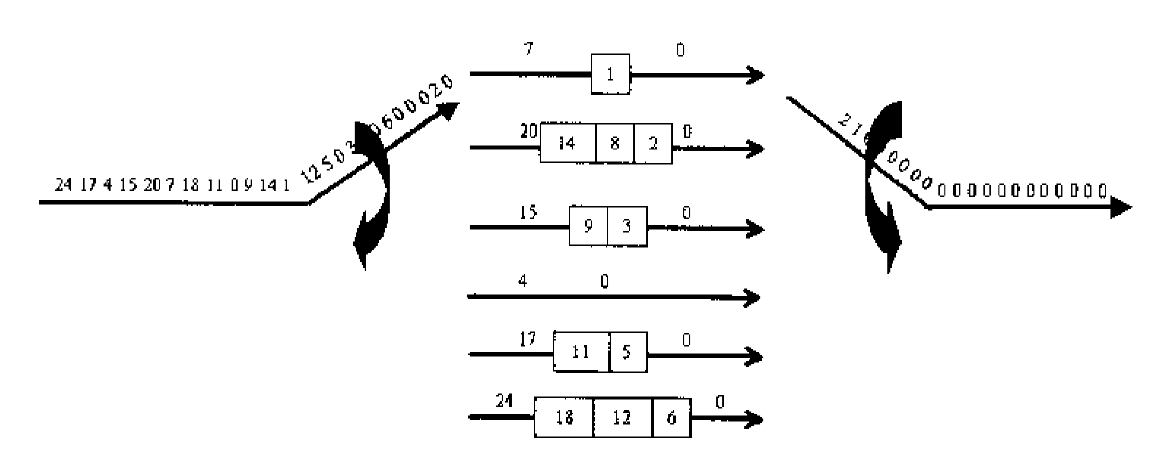



图 7-65　一个复用解交织器的示例

复用解交织器的输入信号 0,2,0,0,6,0,8,3,0,5,12,1,14,9… 就是图 7-63 中的复用交织器的输出信号，解交织的过程与交织过程类似，复用解交织器的开关依次切换到各行移位寄存器，输入数据存放在最左端的寄存器中，同时输出最右端寄存器的数据。从图 7-65 可以看到，在输出第一个解交织数据之前，共输出了 18 个 0（即复用交织器的初始状态），因此，数据在解交织之后将引入一定的时延，时延的大小取决于复用交织器的内部结构。复用解交织器模块有以下两个参数。

■ Interleaver delay (samples)（交织延时）

复用解交织器的交织时延列向量 $L=[L_1,L_2,\cdots L_N]'$，向量的长度等于复用交织器中移位寄存器的行数 $N$，向量中的每个元素表示该行移位寄存器中寄存器的数目。本参数应该与产生交织信号的复用交织器的交织时延向量相同。当输入信号是抽样信号时，本参数也可以是一个行向量。例如，图 7-65 所示的复用交织器的交织时延等于 [2,0,1,3,1,0]′。

■ Initial conditions（初始状态）

本参数表示与复用解交织器相对应的复用交织器移位寄存器的初始状态。当本参数是一个标量时，复用交织器中所有寄存器的初始状态都等于这个数值；当本参数设置位一个向量时，向量的长度与交织时延向量的长度相同，其中每一个元素对应于这一行的移位寄存器的初始状态（同一行移位寄存器中的各个寄存器具有相同的初始状态）。当交织时延向量中的某个元素等于 0 时，初始状态向量中与之对应的元素的数值不起作用。本参数应该与产生交织信号的复用交织器的初始状态参数相同。

## 7.5.2　卷积交织

卷积交织器是一类特殊的复用交织器，它的工作原理与复用交织器相同，两者的差别在于内部移位寄存器数目的设置方式。在复用交织器中，每一行移位寄存器的数目都可以由用户自己设置，而在卷积交织器中，相邻两行的移位寄存器数目的差值是固定的，且第一行移位寄存器的数目为 0。本节将依次介绍卷积交织器和卷积解交织器的原理及其使用方法。

### 1. 卷积交织器

卷积交织器（Convolutional Interleaver）是复用交织器的一种特殊形式，它采用与复用交织器相同的原理对输入信号实施交织。卷积交织器模块的输入信号可以是标量，也可以是帧格式的列向量，其中每个元素可以是实信号，也可以是复信号。卷积交织器模块的输出信号的抽样间隔与输入信号相同。图 7-66 所示是卷积交织器模块及其参数设置对话框。

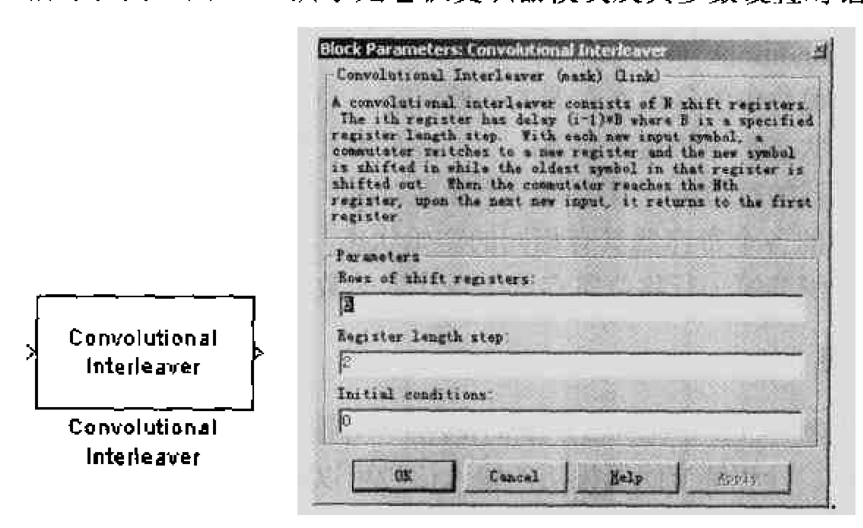



图 7-66　卷积交织器模块及其参数设置对话框

卷积交织器的工作原理与复用交织器相同，它们的差别表现在每一行移位寄存器的数目上。卷积交织器由 $N$ 个并行的移位寄存器组成，其中第 $i(1 \leqslant i \leqslant N)$行移位寄存器有$(i-1) \times K$ 个寄存器，其中 $K$ 是卷积交织器的寄存器长度间隔。图 7-67 所示是当 $N$ 等于 3，$K$ 等于 2 时卷积交织器的结构。

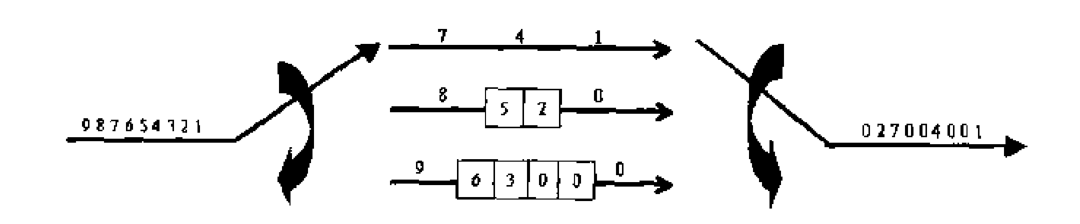



图 7-67　一个卷积交织器的示例

在这个卷积交织器中，第一行没有寄存器，第二行有两个寄存器，第三行有 4 个寄存器，因此，如果输入信号进入第一行的移位寄存器，它直接进入输出序列；如果输入信号进入第二行或第三行移位寄存器，它的输出时延分别是 2 个和 4 个抽样周期。在初始状态，

所有寄存器的数值都等于 0，这个初始状态是通过参数 Initial conditions 设置的。假设输入信号依次为1,2,3…，在第一个抽样周期内，输入信号 1 进入第一行移位寄存器，它直接产生输出；在第二个抽样周期，卷积交织器的两个开关同时切换到第二行移位寄存器，输出最右端寄存器中的数据 0，同时把输入信号 2 保存在最左端的寄存器中。同理，当卷积交织器的开关切换到第三行移位寄存器时，卷积交织器已经产生的输出数据是1,0,0 。在第四个抽样周期到来的时候，卷积交织器的开关重新切换到第一行移位寄存器。卷积交织器按照上面的过程循环工作，从而实现对输入数据的交织。卷积交织器模块有以下 3 个参数。

- Rows of shift registers Interleaver delay (samples)（移位寄存器的行数）

  卷积交织器中移位寄存器的行数 $N$。

- Register length step（寄存器长度间隔）

  卷积交织器中各个移位寄存器的长度间隔 $K$，它表示相邻两行的移位寄存器中的寄存器数目的差值。

- Initial conditions（初始状态）

  卷积交织器模块中的移位寄存器的初始状态。当本参数是一个标量时，卷积交织器中所有寄存器的初始状态都等于这个数值；当本参数设置位一个向量时，向量的长度等于卷积交织器中移位寄存器的行数 $N$，其中每一个元素对应于这一行的移位寄存器的初始状态（同一行移位寄存器中的各个寄存器具有相同的初始状态）。初始状态向量中的第一个元素不起作用，因为卷积交织器第一行移位寄存器中寄存器的数目为 0。

## 2. 卷积解交织器

卷积解交织器（Convolutional Deinterleaver）对输入的卷积交织信号实施解交织，还原得到交织前的数据。卷积解交织器模块的输入信号可以是标量，也可以是帧格式的列向量，其中每个元素可以是实信号，也可以是复信号。卷积解交织器模块的输出信号的抽样间隔与输入信号相同。卷积解交织器及其参数设置对话框如图 7-68 所示。

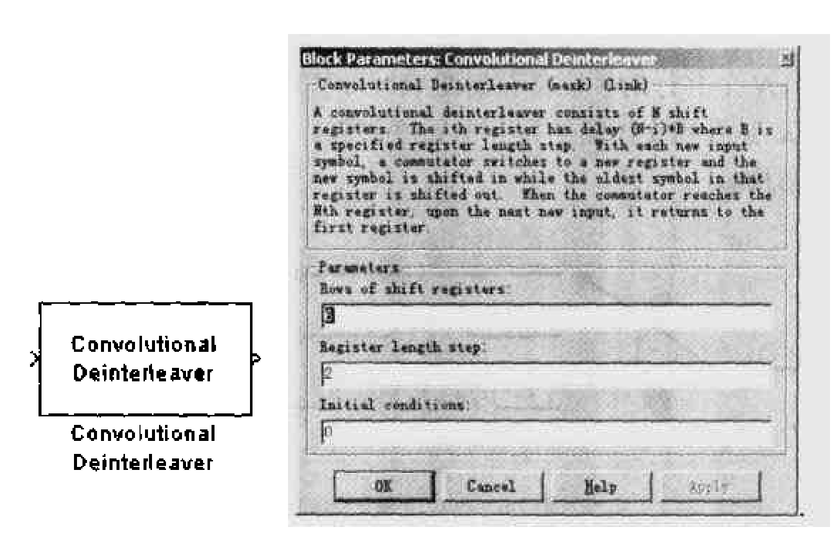



图 7-68　卷积解交织器及其参数设置对话框

卷积解交织器的工作原理与复用解交织器相同。在卷积交织器中，它有 $N$ 个并行的移位寄存器。当卷积解交织器的寄存器长度间隔等于 $K$ 时，第 $i(1\leq i\leq N)$行移位寄存器有$(N-i)\times K$ 个

寄存器。图 7-69 所示是与图 7-67 相对应的卷积解交织器，其中 $N$ 等于 3，$K$ 等于 2。

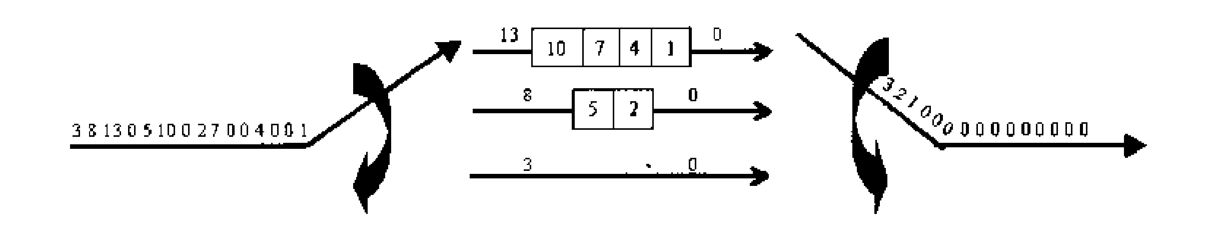



图 7-69　一个卷积解交织器的示例

在图 7-69 所示中，卷积解交织器的输入信号 1,0,0,4,0,0,7,2,0… 是相应的卷积交织器的输出信号。卷积解交织器的开关依次切换到各行移位寄存器，输入数据存放在最左端的寄存器中，同时输出最右端寄存器的数据。从图中可以看到，在输出第一个解交织数据 1 之前，共输出了 12 个 0（即卷积交织器的初始状态）。卷积交织信号在解交织之后将会产生一定的时延，时延的大小取决于卷积交织器的构造。卷积解交织器有以下 3 个参数。

■　Rows of shift registers Interleaver delay (samples)（移位寄存器的行数）

卷积解交织器中移位寄存器的行数 $N$。

■　Register length step（寄存器长度间隔）

卷积解交织器中各个移位寄存器的长度间隔 $K$，它表示相邻两行的移位寄存器中的寄存器数目的差值。

■　Initial conditions（初始状态）

本参数表示与卷积解交织器相对应的卷积交织器移位寄存器的初始状态。当本参数是一个标量时，卷积交织器中所有寄存器的初始状态都等于这个数值；当本参数设置位一个向量时，向量的长度等于卷积交织器中移位寄存器的行数 $N$，其中每一个元素对应于这一行的移位寄存器的初始状态（同一行移位寄存器中的各个寄存器具有相同的初始状态）。初始状态向量中的第一个元素不起作用，因为卷积交织器第一行移位寄存器中寄存器的数目为 0。

## 7.5.3　螺旋交织

螺旋交织把输入信号按行放置在一个矩阵中，然后按照螺旋顺序读出数据，产生交织信号。螺旋交织可以分成两个步骤：首先通过矩阵交织把输入信号转换成块交织信号，然后通过复用交织得到螺旋交织信号。MATLAB 中的螺旋交织器模块就是按照这种方式工作的。本节我们将依次介绍螺旋交织器和螺旋解交织器。

### 1. 螺旋交织器

螺旋交织器（Helical Interleaver）采用一个 C 列的矩阵 M 对输入信号实施螺旋交织。在螺旋交织器中，矩阵 M 的行数是无穷的，它可以按照输入信号的长度自动增加。螺旋交织器的输入信号时一个向量（列向量或行向量），向量的长度等于 C×N，其中 $N$ 是螺旋交织器的分组长度。当输入信号是帧格式的数据时，它只能是列向量。螺旋交织器及其参数设置对话框如图 7-70 所示。

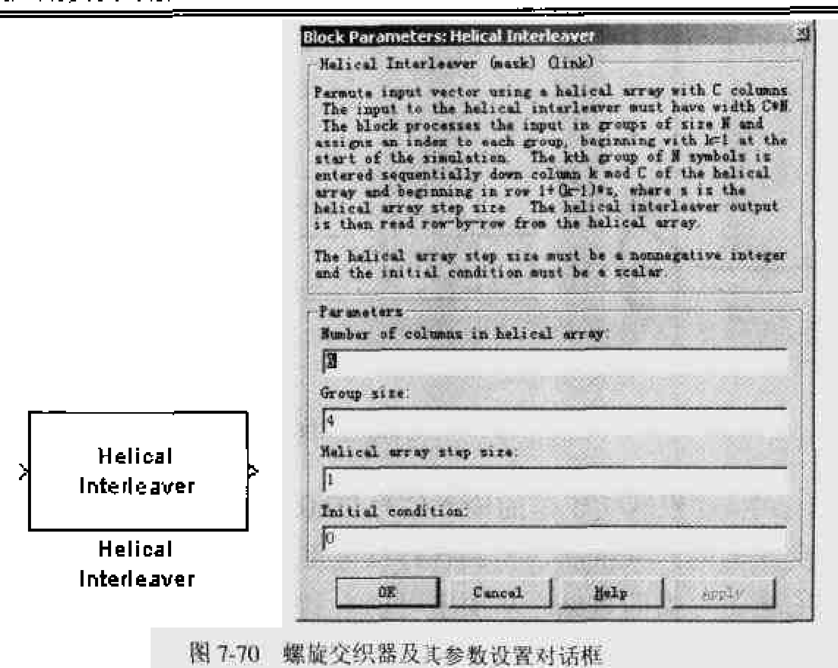



图 7-70 螺旋交织器及其参数设置对话框

螺旋交织器把长度为 $C\times N$ 的输入信号分成 $C$ 组，每组长度为 $N$。在初始状态，螺旋交织器矩阵 $M$ 中的元素都等于初始值，这个初始值可以通过 Initial condition 设置。螺旋交织器把每组长度为 $N$ 的信号按照螺旋方式分别放置在矩阵 $M$ 中，其中第一组的元素依次放在矩阵 $M$ 的第一列，第 $i(1\leqslant i\leqslant C)$组的元素放在第 $i$ 列，以此类推。假设矩阵 $M$ 的第一列中具有最小行号的空白单元（即该单元的数值等于初始值）的行号为 $x$，则对于第 $k$ 组中的第一个元素，它被放置于矩阵 $M$ 的第 $x+(k-1)\times s$ 行和第 $k$ 列，其中 $s$ 是螺旋交织的步长参数（Helical array step size）。第 $k$ 组的其他元素则依次放置在同一列中。

螺旋交织器每次从矩阵 $M$ 中依次读取 $N$ 行元素作为输出信号。在某一时刻，输出信号中可能包含了初始状态值，也包含了该时刻之前以及当前时刻的输入信号。图 7-71 所示是当 $C$ 等于 3 以及 $N$ 等于 2 时的螺旋交织器，其中螺旋交织的步长 $s$ 等于 1。

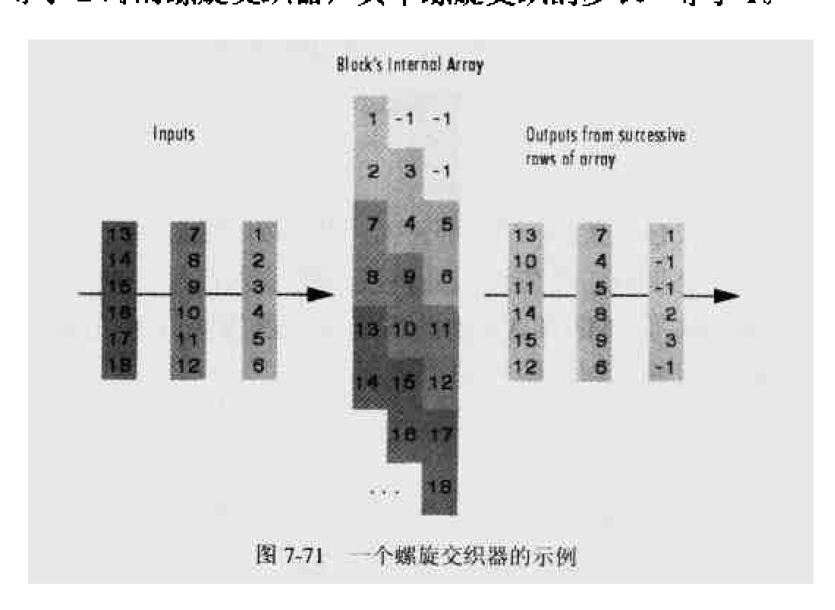



图 7-71 一个螺旋交织器的示例

图 7-72 所示是螺旋交织器的内部结构。从图 7-72 所示中可以看到，螺旋交织器实际上是由矩阵交织器和复用交织器组成的，输入信号首先通过矩阵交织器实施交织，然后再通过一个复用交织器，从而得到螺旋交织信号。

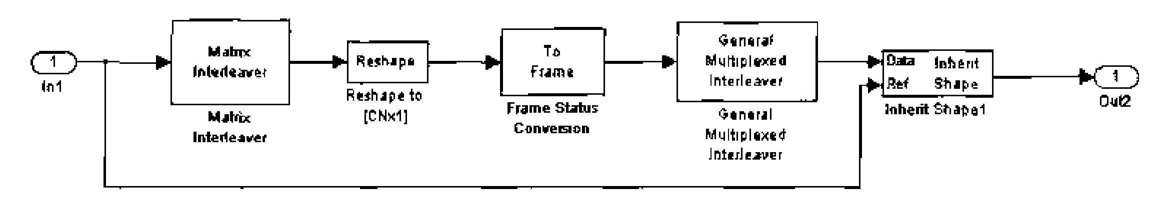



图 7-72　螺旋交织器的内部结构

螺旋交织器主要有以下几个参数。

■　Number of columns in helical array（矩阵的列数）

螺旋交织器中矩阵 $M$ 的列数 $C$。

■　Group size（分组长度）

螺旋交织器的分组长度 $N$。

■　Helical array step size（交织步长）

螺旋交织的步长 $s$。

■　Initial condition（初始状态）

螺旋交织器中矩阵 $M$ 的初始状态。本参数是一个标量。

### 2．螺旋解交织器

螺旋解交织器（Helical Deinterleaver）对螺旋交织信号实施解交织，还原得到交织前的信号。螺旋解交织器按照行顺序把输入信号依次放入一个 $C$ 列的矩阵 $M$，然后按照螺旋顺序依次读出信号。如果与螺旋解交织器相对应的螺旋交织器的分组长度是 $N$，则螺旋交织器的输入信号时一个长度等于 $C\times N$ 向量（列向量或行向量）。当输入信号是帧格式的数据时，它只能是列向量。螺旋解交织器模块及其参数设置对话框如图 7-73 所示。

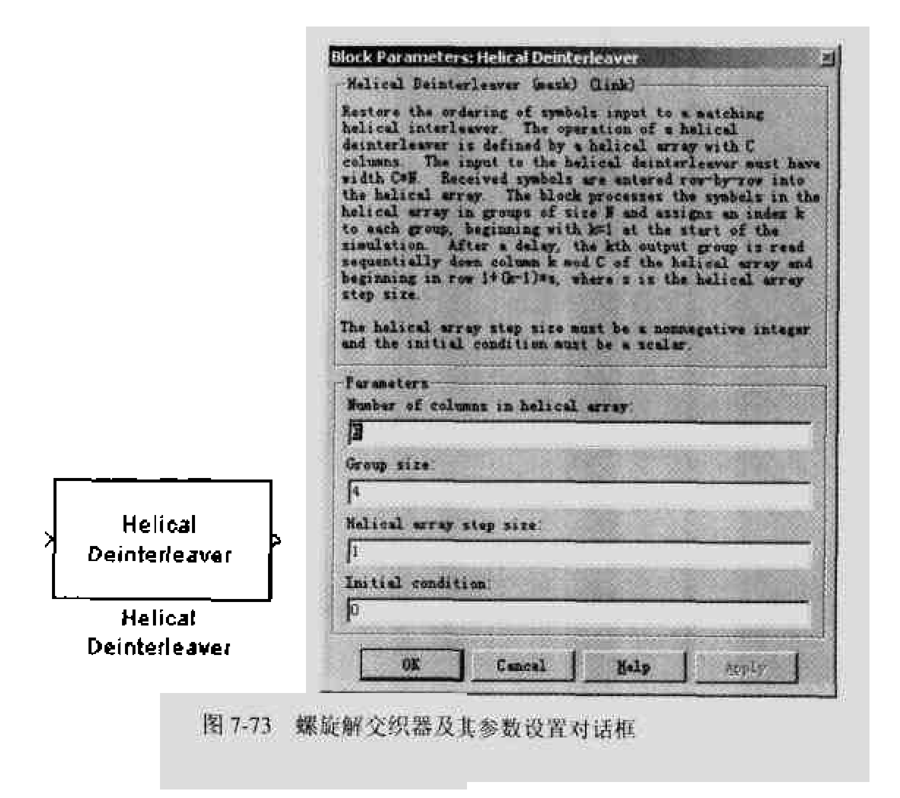



图 7-73　螺旋解交织器及其参数设置对话框

在初始状态，螺旋解交织器矩阵 $M$ 中的元素都等于初始值，这个初始值可以通过 Initial condition 设置，并且这个初始值可以与产生输入信号的螺旋交织器的初始值不同。螺旋交织器把长度等于 $C\times N$ 的输入信号依次放置在矩阵 $M$ 的 $N$ 个空白行中，并且从矩阵 $M$ 的第一列开始每列读取 $N$ 个元素作为输出信号。如果矩阵 $M$ 的第一列中具有最小行号的已读取单元的行号为 $x$，螺旋交织的步长为 $s$，则对于第 $k$ 列元素，螺旋解交织器从矩阵 $M$ 的第 $x+(k-1)\times s$ 行和第 $k$ 列开始依次读取 $N$ 个输出信号。

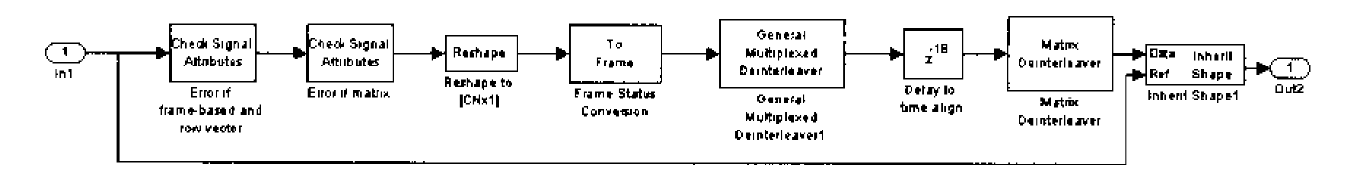



图 7-74 螺旋解交织器的内部结构

图 7-74 所示是螺旋交织器的内部结构。从图中可以看到，螺旋交织器实际上是由矩阵解交织器和复用解交织器组成的，输入的螺旋交织信号首先通过复用解交织器实施解交织，然后再通过一个矩阵解交织器，最后得到相应的解交织信号。

螺旋解交织器主要有以下几个参数。

■ Number of columns in helical array（矩阵的列数）

与螺旋解交织器相对应的螺旋交织器中矩阵 $M$ 的列数 $C$。

■ Group size（分组长度）

与螺旋解交织器相对应的螺旋交织器的分组长度 $N$。

■ Helical array step size（交织步长）

与螺旋解交织器相对应的螺旋交织的步长 $s$。

■ Initial condition（初始状态）

螺旋解交织器中矩阵 $M$ 的初始状态。本参数是一个标量。

# 第 8 章　信号调制

带宽是移动通信系统中最紧缺的资源，因此，如何在有限的带宽里获得足够高的传输速率，即如何获得比较高的频率利用率是移动通信系统中的一个关键问题。信号调制是根据输入信号改变传输信号，使之能够在特定的频率范围之内和特定条件的信道中传输的过程。信号解调是信号调制的逆过程，它把具有某种特定形态的传输波形还原为发送端调制前的信号。

根据调制前的信号是模拟信号还是数字信号可以把信号调制方式分成两类：模拟调制方式和数字调制方式。模拟调制方式还可以分成模拟幅度调制、模拟频率调制和模拟相位调制，这 3 种调制方式分别根据输入的模拟信号的强度改变载频信号的幅度、频率和相位。对于数字调制方式，它也存在着上述的 3 种调制方式：幅移键控、频移键控和相移键控。本章后面的内容将分别介绍这几种调制方式及相应的解调方式。

## 8.1　模拟幅度调制

模拟调制是信号的幅度、频率或相位随着模拟输入信号的变化而相应地发生变化的调制方式，它包括幅度调制、频率调制和相位调制 3 种，其中幅度调制由可以分成双边带幅度调制、双边带抑制载波调制以及单边带调制。另外，根据调制信号是否存在载波还可以把模拟调制方式分成基带调制和频带调制，其中基带调制方式不需要载波，而频带调制方式则把基带信号调制到更高频率的载波上进行传输。本节依次介绍这几种调制方式的原理及其使用。

### 8.1.1　双边带幅度调制

双边带幅度调制包括双边带基带幅度调制（Double-SideBand Amplitude Modulation Baseband）和双边带频带幅度调制（Double-SideBand Amplitude Modulation Passband）。假设双边带幅度调制的输入信号是$u(t)$，输出信号是 $y(t)$，则在 MATLAB 中，对于双边带基带幅度调制，$y(t)$ 是个复信号，且 $y(t)=(u(t)+k)e^{j\theta}$；而对于双边带频带幅度调制，$y(t)$ 是个实信号，且 $y(t)=(u(t)+k)\cos(2\pi f_c t+\theta)$。本节将介绍双边带基带幅度调制器和双边带频带幅度调制器。

**1. 双边带基带幅度调制**

双边带基带幅度调制器（DSB AM Modulator Baseband）实现基带模拟信号的双边带幅度调制。双边带基带幅度调制器的输入信号是实数形式的标量信号，输出信号则是复数形式的信号。双边带基带幅度调制器模块及其参数设置对话框如图 8-1 所示。

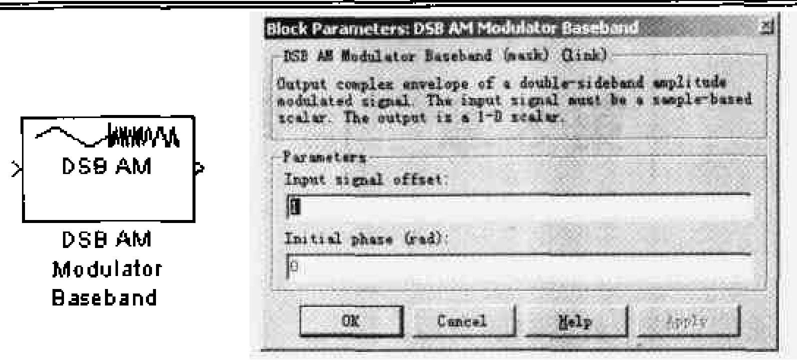



图 8-1　双边带基带幅度调制器模块及其参数设置对话框

假设双边带基带幅度调制器的输入信号是$u(t)$，则输出信号等于$y(t)=(u(t)+k)e^{j\theta}$，其中$k$表示输入信号偏移，它对应于参数 Input signal offset；$\theta$表示初始相位，它对应于参数 Initial phase (rad)。双边带基带幅度调制器主要有以下几个参数。

■　Input signal offset（输入信号偏移）

输入信号偏移$k$。本参数应该大于或等于输入信号最小值的绝对值。

■　Initial phase (rad)（初始相位）

本参数表示双边带基带幅度调制信号的初始相位。

双边带基带幅度解调器（DSB AM Demodulator Baseband）对双边带基带幅度调制信号进行解调。双边带基带幅度解调器的输入信号是一个复数形式的抽样信号，解调后的输出信号是一个实信号。双边带基带幅度解调器模块及其参数设置对话框如图 8-2 所示。

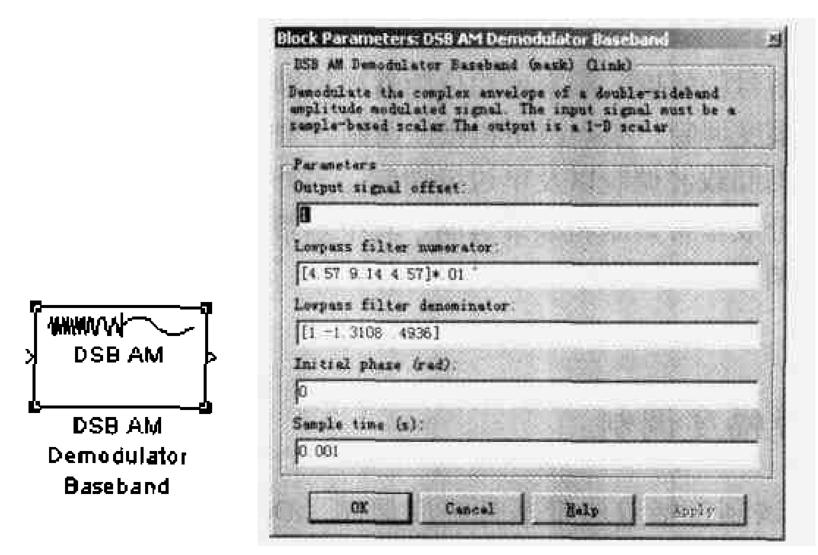



图 8-2　双边带基带幅度解调器模块及其参数设置对话框

在解调过程中，双边带基带幅度解调器使用了低通滤波器，这个低通滤波器的转换函数由参数 Lowpass filter numerator 和 Lowpass filter denominator 确定，它们分别表示低通滤波器转换函数的分子和分母。双边带基带幅度解调器模块主要有以下几个参数。

■　Output signal offset（输出信号偏移）

本参数表示双边带基带幅度解调器的输出信号偏移，双边带基带幅度解调器的解调信号都将减去这个偏移值，从而得到输出数据。

■　Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低

的顺序表示低通滤波器转换函数的系数的分子。

■　Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

■　Initial phase (rad)（初始相位）

本参数等于与双边带基带幅度解调器相对应的调制器的初始相位 $\theta$。

■　Sample time（抽样间隔）

双边带基带幅度解调器输出的解调信号的抽样间隔。

**2. 双边带频带幅度调制**

双边带频带幅度调制器（DSB AM Modulator Passband）对输入的模拟信号实施双边带幅度调制，产生频带幅度调制信号。双边带频带幅度调制器的输入信号和输出信号都是抽样形式的实信号。双边带频带幅度调制器模块及其参数设置对话框如图 8-3 所示。

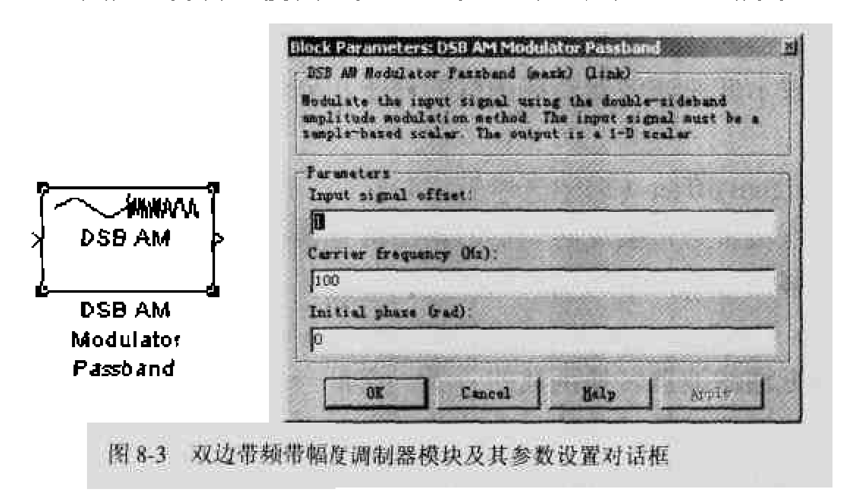



图 8-3　双边带频带幅度调制器模块及其参数设置对话框

假设双边带频带幅度调制的输入信号是 $u(t)$，输出信号是 $y(t)$，则

$$y(t) = (u(t) + k)\cos(2\pi f_c t + \theta) \tag{8.1}$$

其中 $k$ 是输入信号偏移，$f_c$ 是载波频率，$\theta$ 是初始相位。双边带频带幅度调制器主要有以下几个参数。

■　Input signal offset（输入信号偏移）

输入信号偏移 $k$。本参数应该大于或等于输入信号最小值的绝对值。

■　Carrier frequency (Hz)（载波频率）

双边带频带幅度调制信号的载波频率（单位：Hz）。

■　Initial phase (rad)（初始相位）

本参数表示双边带频带幅度调制信号的初始相位（单位：弧度）。

双边带频带幅度解调器（DSB AM Demodulator Passband）对双边带频带幅度调制基带信号进行解调，它的输入信号和输出信号都是一个抽样的实信号。双边带频带幅度解调器模块及其参数设置对话框如图 8-4 所示。

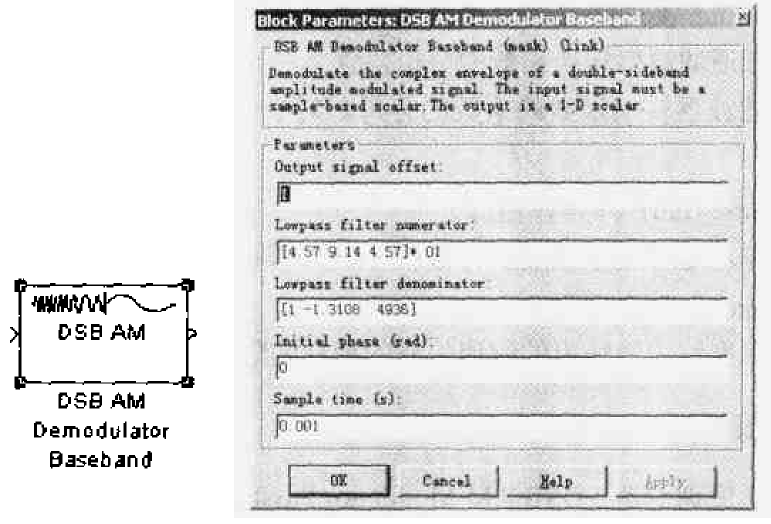



图 8-4 双边带频带幅度解调器模块及其参数设置对话框

双边带频带幅度解调器通过包络检测对信号进行解调，并且使用低通滤波器实现滤波功能，这个低通滤波器的转换函数由参数 Lowpass filter numerator 和 Lowpass filter denominator 确定，它们分别表示低通滤波器转换函数的分子和分母。双边带频带幅度解调器模块主要有以下几个参数。

■ Output signal offset（输出信号偏移）

本参数表示双边带频带幅度解调器的输出信号偏移，双边带频带幅度解调器的解调信号都将减去这个偏移值，从而得到输出数据。

■ Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

■ Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

■ Initial phase (rad)（初始相位）

本参数等于与双边带频带幅度解调器相对应的调制器的初始相位 $\theta$。

■ Sample time（抽样间隔）

双边带频带幅度解调器输出的解调信号的抽样间隔。

## 8.1.2 双边带抑制载波幅度调制

双边带抑制载波幅度调制也包括基带和频带两种。假设输入信号是 $u(t)$，输出信号是 $y(t)$，对于双边带基带抑制载波幅度调制，$y(t)$ 是个复信号，且 $y(t)=u(t)e^{j\theta}$；而对于双边带频带抑制载波幅度调制，$y(t)$ 是个实信号，且 $y(t)=u(t)\cos(2\pi f_c t+\theta)$。对比双边带抑制载波幅度调制和双边带幅度调制的公式可以看到，双边带抑制载波幅度调制实际上是双边带幅度调制当 $k=0$ 时的一个特例。

### 1. 双边带基带抑制载波幅度调制

双边带基带抑制载波幅度调制器（DSBSC AM Modulator Baseband）实现基带模拟信号的双边带抑制载波幅度调制。双边带基带抑制载波幅度调制器的输入信号是实数形式的标量信号，输出信号则是复数形式的信号，其模块框图及参数设置对话框如图 8-5 所示。

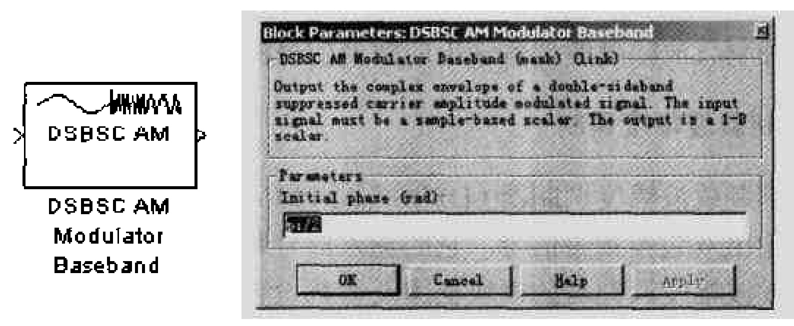



图 8-5 双边带基带抑制载波幅度调制器模块及其参数设置对话框

假设双边带基带抑制载波幅度调制器的输入信号是 $u(t)$，则输出信号等于 $u(t)e^{j\theta}$，其中 $\theta$ 表示初始相位 Initial phase。双边带基带抑制载波幅度调制器只有一个参数。

■ Initial phase (rad)（初始相位）

本参数表示双边带基带抑制载波幅度调制信号的初始相位。

双边带基带抑制载波幅度解调器（DSBSC AM Demodulator Baseband）对双边带基带抑制载波幅度调制基带信号进行解调。双边带基带抑制载波幅度解调器的输入信号是一个复数形式的抽样信号，解调后的输出信号是一个实信号，其模块框图及参数设置对话框如图 8-6 所示。

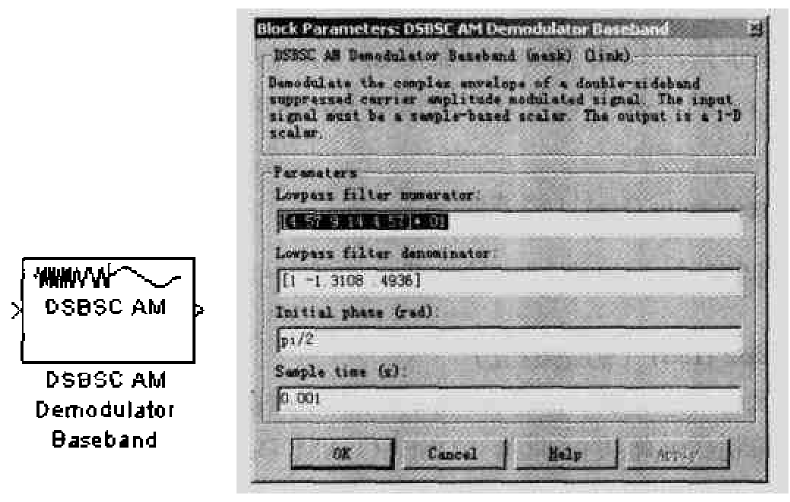



图 8-6 双边带基带抑制载波幅度解调器模块及参数设置对话框

在解调过程中，双边带基带抑制载波幅度解调器使用了低通滤波器，这个低通滤波器的转换函数由参数 Lowpass filter numerator 和 Lowpass filter denominator 确定，它们分别表示低通滤波器转换函数的分子和分母。双边带基带抑制载波幅度解调器模块主要有以下几个参数。

■ Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

■ Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低

的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

■ Initial phase (rad)（初始相位）

本参数等于与双边带基带抑制载波幅度解调器相对应的调制器的初始相位 $\theta$。

■ Sample time（抽样间隔）

双边带基带抑制载波幅度解调器输出的解调信号的抽样间隔。

## 2. 双边带频带抑制载波幅度调制

双边带频带抑制载波幅度调制器（DSBSC AM Modulator Passband）对输入信号实施抑制载波幅度调制，输出双边带频带信号，它的输入信号和输出信号都是复数形式的抽样信号。双边带频带抑制载波幅度调制器的模块框图及参数设置对话框如图 8-7 所示。

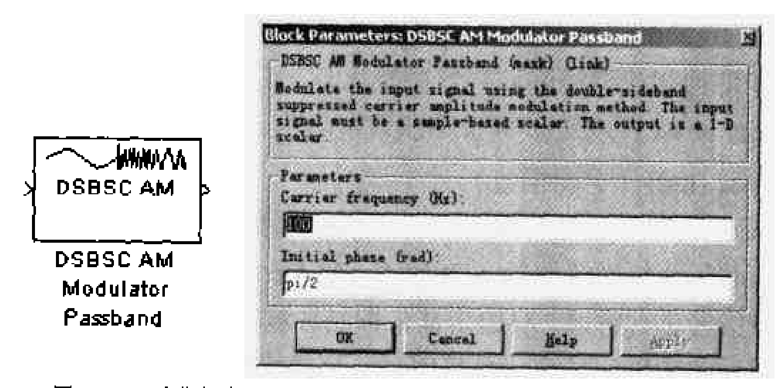



图 8-7 双边带频带抑制载波幅度调制器模块及其参数设置对话框

假设双边带频带抑制载波幅度调制器的输入信号是 $u(t)$，输出信号为 $y(t)$，则两者满足 $y(t)=u(t)\cos(2\pi f_c t+\theta)$，其中 $\theta$ 表示初始相位 Initial phase。双边带频带抑制载波幅度调制器有两个参数。

■ Carrier frequency (Hz)（载波频率）

双边带频带抑制载波幅度调制信号的载波频率 $f_c$（单位：Hz）。

■ Initial phase (rad)（初始相位）

本参数表示双边带频带抑制载波幅度调制信号的初始相位 $\theta$（单位：弧度）。

双边带频带抑制载波幅度解调器（DSBSC AM Demodulator Passband）对双边带频带抑制载波幅度调制基带信号进行解调，它的输入信号和输出信号都是实数形式的抽样信号，其模块框图及参数设置对话框如图 8-8 所示。

在解调过程中，双边带频带抑制载波幅度解调器使用的低通滤波器的转换函数由参数 Lowpass filter numerator 和 Lowpass filter denominator 确定，它们分别表示低通滤波器转换函数的分子和分母。双边带频带抑制载波幅度解调器模块主要有以下几个参数。

■ Carrier frequency (Hz)（载波频率）

双边带频带抑制载波幅度调制信号的载波频率（单位：Hz）。

■ Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

■　Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

■　Initial phase (rad)（初始相位）

本参数等于与双边带频带抑制载波幅度解调器相对应的调制器的初始相位 $\theta$（单位：弧度）。

■　Sample time（抽样间隔）

双边带频带抑制载波幅度解调器输出的解调信号的抽样间隔。

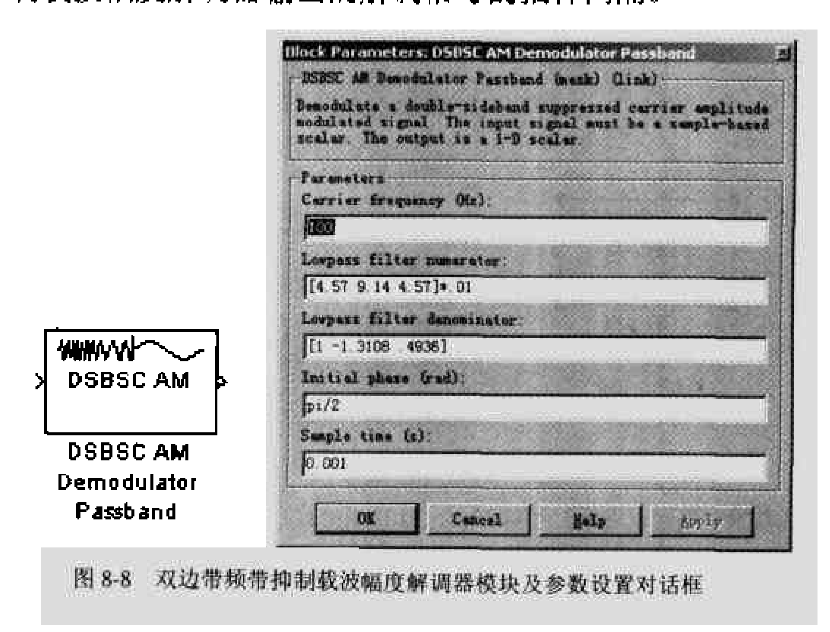



图 8-8　双边带频带抑制载波幅度解调器模块及参数设置对话框

## 8.1.3　单边带幅度调制

在单边带幅度调制 SSB AM（Single-SideBand Amplitude Modulation）中，发送端只传输基带幅度调制信号的上边带或下边带，它使用的带宽只有双边带调制信号的一半，因而具有更高的频谱利用率。本节介绍单边带基带幅度调制和单边带频带幅度调制。

### 1. 单边带基带幅度调制

单边带基带幅度调制器（SSB AM Modulator Baseband）对实数形式的抽样信号进行单边带幅度调制，产生一个复数形式的输出信号。假设输入信号是 $u(t)$，输出信号为 $y(t)$，且它们之间满足关系式 $y(t)=(u(t)\pm j\hat{u}(t))e^{j\theta}$，其中$\theta$是初始相位，$\hat{u}(t)$ 是输入信号 $u(t)$的希尔伯特变换，加号表示输出上边带信号，减号则表示输出下边带信号。单边带基带幅度调制器模块及其参数设置对话框如图 8-9 所示。

单边带基带幅度调制器模块有以下几个参数。

■　Initial phase (rad)（初始相位）

单边带基带幅度调制信号的初始相位$\theta$。

■　Bandwidth of the input signal (Hz)（输入信号带宽）

单边带基带幅度调制模块输入信号的最高频率（单位：Hz）。

■ Time delay for Hilbert transform filter (s)（希尔伯特变换滤波器的时延）

单边带基带幅度调制模块中的希尔伯特变换滤波器的时延。

■ Sample time（抽样间隔）

单边带基带幅度调制模块中的希尔伯特变换滤波器的抽样间隔。

■ Sideband to modulate（调制边带）

本参数可以是 Upper 或 Lower，用于确定单边带基带幅度调制的上边带或下边带。

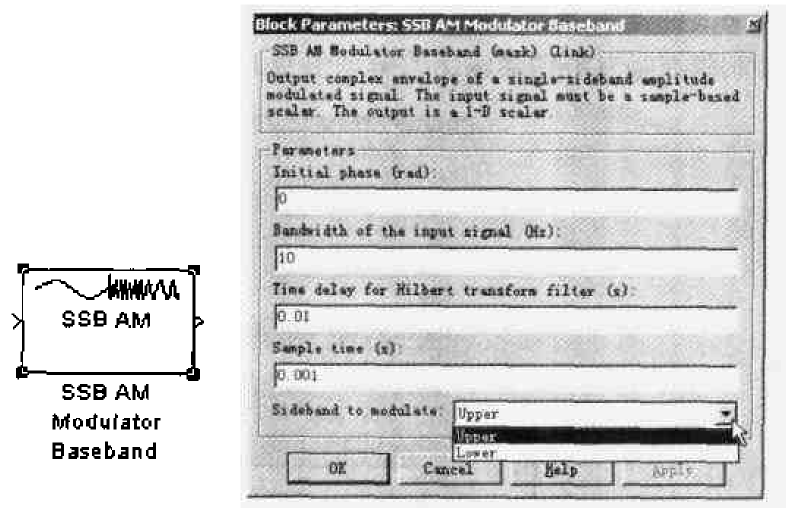



图 8-9 单边带幅度调制器模块及其参数设置对话框

单边带基带幅度解调器（SSB AM Demodulator Baseband）把输入的单边带基带调制复信号还原为初始的实信号。单边带基带幅度解调器模块及其参数设置对话框如图 8-10 所示。

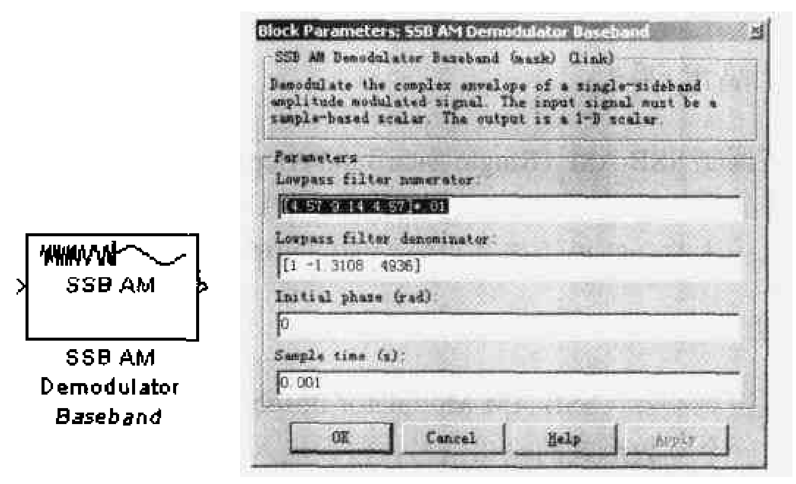



图 8-10 单边带幅度解调器模块及其参数设置对话框

在解调过程中单边带基带幅度解调器使用了一个低通滤波器，这个低通滤波器由 Lowpass filter numerator 和 Lowpass filter denominator 两个参数确定。单边带基带幅度解调器主要有以下几个参数。

■ Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

■ Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺

序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

■　Initial phase (rad)（初始相位）

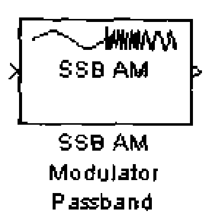
本参数等于与单边带基带幅度解调器相对应的调制器的初始相位 $\theta$（单位：弧度）。

■　Sample time（抽样间隔）

单边带基带幅度解调器输出的解调信号的抽样间隔（单位：秒）。

## 2. 单边带频带幅度调制

单边带频带调制器（SSB AM Modulator Passband）对实数形式的抽样信号进行单边带幅度调制，产生的输出信号也是一个实信号。假设输入信号是 $u(t)$，输出信号为 $y(t)$，则它们之间满足关系式 $y(t)=u(t)\cos(f_c t+\theta)\mp\hat{u}(t)\sin(f_c t+\theta)$，其中 $f_c$ 是载波频率，$\theta$ 是初始相位，$\hat{u}(t)$ 是输入信号 $u(t)$ 的希尔伯特变换，减号表示输出上边带信号，加号则表示输出下边带信号。单边带频带幅度调制器模块及其参数设置对话框如图 8-11 所示。



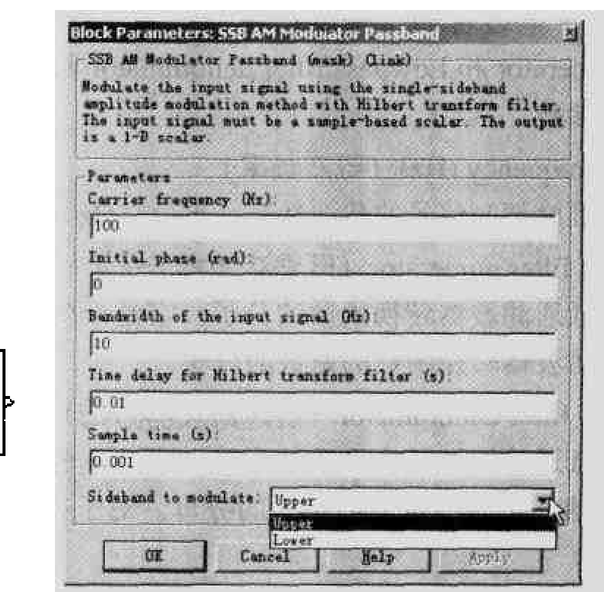




图 8-11　单边带频带幅度调制器模块及其参数设置对话框

单边带频带幅度调制器模块有以下几个参数。

■　Carrier frequency (Hz)（载波频率）

单边带频带幅度调制信号的载波频率（单位：Hz）。

■　Bandwidth of the input signal (Hz)（输入信号带宽）

单边带频带幅度调制模块输入信号的最高频率（单位：Hz）。

■　Time delay for Hilbert transform filter (s)（希尔伯特变换滤波器的时延）

单边带频带幅度调制模块中的希尔伯特变换滤波器的时延。

■　Sample time（抽样间隔）

单边带频带幅度调制模块中的希尔伯特变换滤波器的抽样间隔。

■　Sideband to modulate（调制边带）

本参数可以是 Upper，也可以是 Lower，用于指定单边带频带幅度调制器的输出信号是上边带信号或下边带信号。

单边带频带幅度解调器（SSB AM Demodulator Passband）把输入的单边带频带调制信号还原为初始的实信号。单边带频带幅度解调器模块及其参数设置对话框如图 8-12 所示。

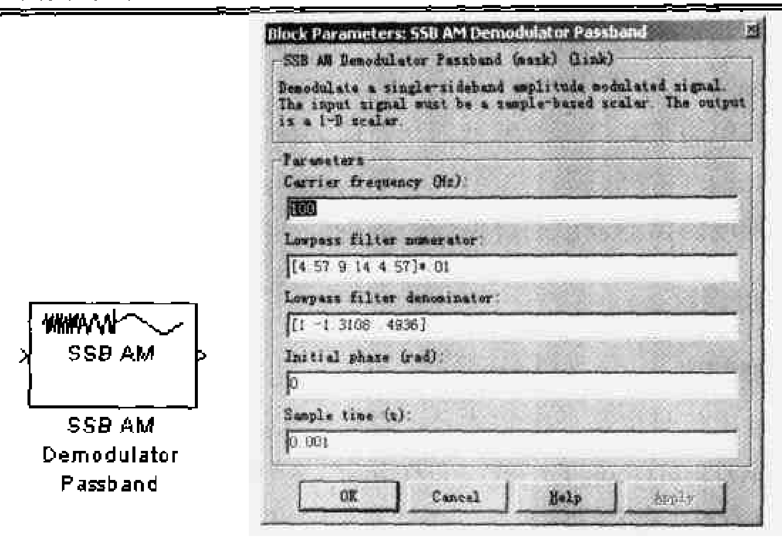



图 8-12　单边带频带幅度解调器模块及其参数设置对话框

在解调过程中单边带频带幅度解调器使用了一个低通滤波器，这个低通滤波器由 Lowpass filter numerator 和 Lowpass filter denominator 两个参数确定。单边带频带幅度解调器主要有以下几个参数。

- Carrier frequency (Hz)（载波频率）

单边带频带幅度调制信号的载波频率（单位：Hz）。

- Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

- Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

- Initial phase (rad)（初始相位）

本参数等于与单边带频带幅度解调器相对应的调制器的初始相位 $\theta$（单位：弧度）。

- Sample time（抽样间隔）

单边带频带幅度解调器输出的解调信号的抽样间隔（单位：秒）。

## 8.2　模拟频率调制

在频率调制（Frequency Modulation）过程中，载波（一般是正弦波）的频率与输入信号的幅度成正比。假设输入信号为 $u(t)$，输出信号为 $y(t)$，则对于基带调制信号，

$$y(t)=\exp(j\theta+2\pi jK_c\int_t u(\tau)d\tau);$$

而对于频带信号，

$$y(t)=\cos(2\pi f_c t+2\pi K_c\int_t u(\tau)d\tau+\theta),$$

其中，$f_c$ 是载波频率，$\theta$是初始相位，$K_c$ 是调制常数。本节将依次介绍基带频率调制和频带频率调制。

## 8.2.1　基带频率调制

基带频率调制器（FM Modulator Baseband）对输入的实信号实施频率调制，产生的复数形式的输出信号。假设输入信号为 $u(t)$，输出信号为 $y(t)$，则输出信号 $y(t)$ 的频率随着输入信号 $u(t)$ 的幅度变化而变化，两者之间满足关系式：

$$y(t) = \exp(j\theta + 2\pi jK_c\int_t u(\tau)d\tau),$$

其中，$\theta$是初始相位，$K_c$ 是调制常数。基带频率调制器模块及其参数设置对话框如图 8-13 所示。

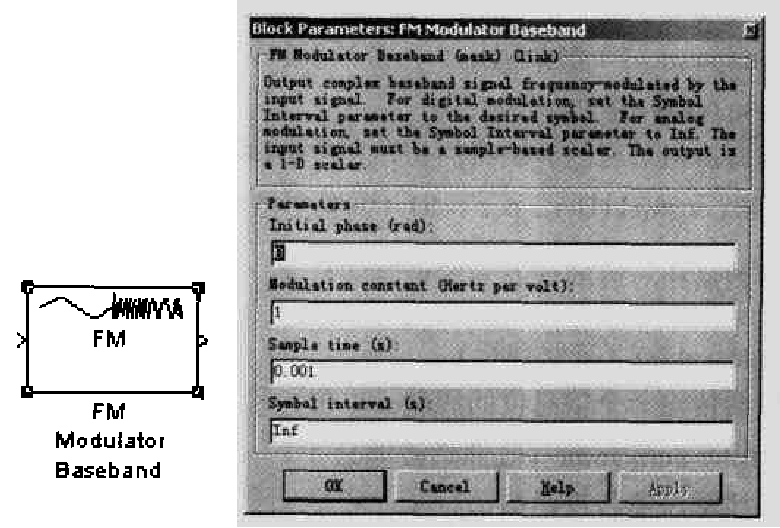



图 8-13　基带频率调制器模块及其参数设置对话框

基带频率调制器模块主要有以下几个参数。

■　Initial phase (rad)（初始相位）

基带频率调制器信号的初始相位$\theta$（单位：弧度）。

■　Modulation constant (Hertz per volt)（调制常数）

基带频率调制器信号的调制常数 $K_c$（单位：Hz/伏特）。

■　Sample time（抽样间隔）

基带频率调制器输出信号的抽样间隔（单位：秒）。

■　Symbol interval (s)（符号间隔）

基带频率调制器传输一个信息位的周期（单位：秒），缺省值是 Inf。

基带频率解调器（FM Demodulator Baseband）从输入的基带频率调制信号中解调出原始的信息序列。基带频率解调器的输入信号是抽样形式的复信号，输出信号是实信号。图 8-14 所示是基带频率解调器模块及其参数设置对话框。

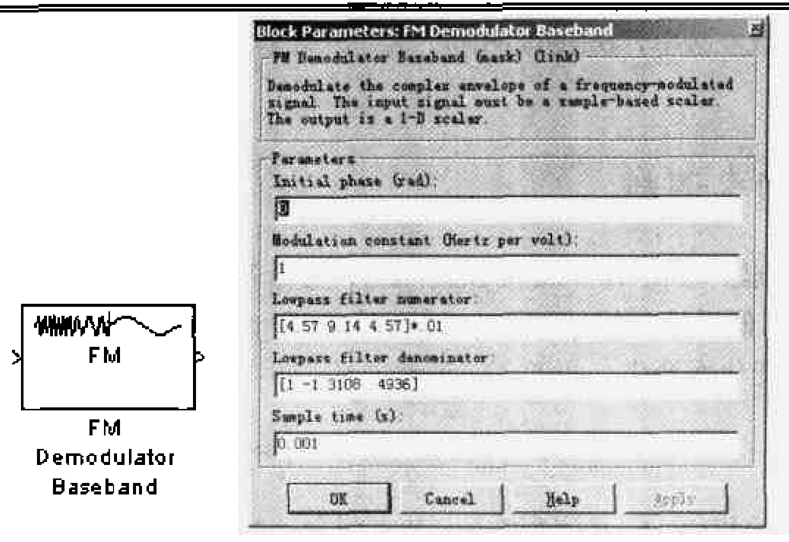



图 8-14　基带频率解调器模块及其参数设置对话框

基带频率解调器使用了一个低通滤波器，这个低通滤波器由 Lowpass filter numerator 和 Lowpass filter denominator 两个参数确定。基带频率解调器主要有以下几个参数。

- Initial phase (rad)（初始相位）

与基带频率解调器相对应的调制器的初始相位$\theta$（单位：弧度）。

- Modulation constant (Hertz per volt)（调制常数）

与基带频率解调器相对应的调制器的调制常数$K_c$（单位：Hz/伏特）。

- Symbol interval (s)（符号间隔）

基带频率调制器传输一个信息位的周期（单位：秒），缺省值是 Inf。

- Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

- Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数被设置为 1。

- Sample time（抽样间隔）

基带频率解调器输出信号的抽样间隔（单位：秒）。

## 8.2.2　频带频率调制

频带频率调制器（FM Modulator Passband）对输入的实信号实施频率调制，产生的抽样形式的调制信号。假设输入信号为$u(t)$，输出信号为$y(t)$，则输入信号和输出信号之间满足关系式：

$$y(t)=\cos(2\pi f_c t+2\pi K_c\int u(\tau)d\tau+\theta),$$

其中$f_c$是载波频率，$\theta$是初始相位，$K_c$是调制常数。频带频率调制器模块及其参数设置对话框如图 8-15 所示。

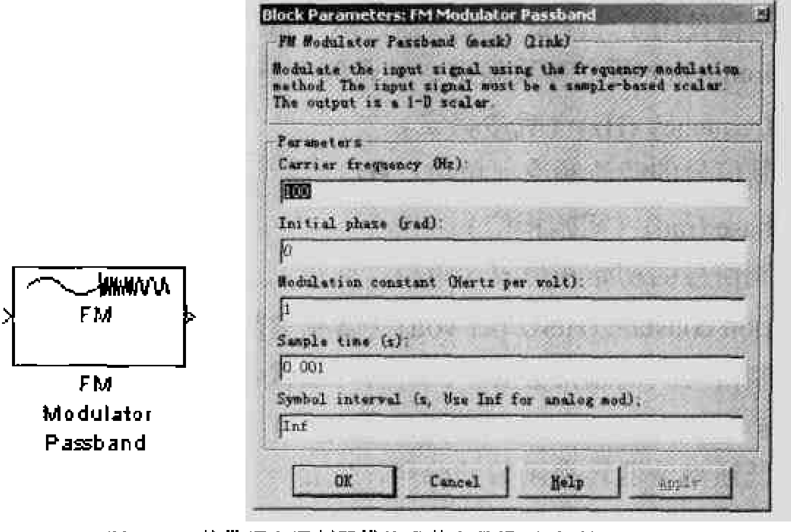



图 8-15　基带频率调制器模块及其参数设置对话框

频带频率调制器模块主要有以下几个参数。

■　Carrier frequency (Hz)（载波频率）

频带频率调制信号的载波频率（单位：Hz）。

■　Initial phase (rad)（初始相位）

频带频率调制器信号的初始相位$\theta$（单位：弧度）。

■　Modulation constant (Hertz per volt)（调制常数）

频带频率调制器信号的调制常数$K_c$（单位：Hz/伏特）。

■　Sample time（抽样间隔）

频带频率调制器输出信号的抽样间隔（单位：秒）。

■　Symbol interval (s)（符号间隔）

频带频率调制器传输一个信息位的周期（单位：秒），缺省值是 Inf。

频带频率解调器（FM Demodulator Passband）从输入的频带频率调制信号中解调出原始的信息序列。频带频率解调器的输入信号是抽样形式的复信号，输出信号是实信号。图 8-16 所示是频带频率解调器模块及其参数设置对话框。

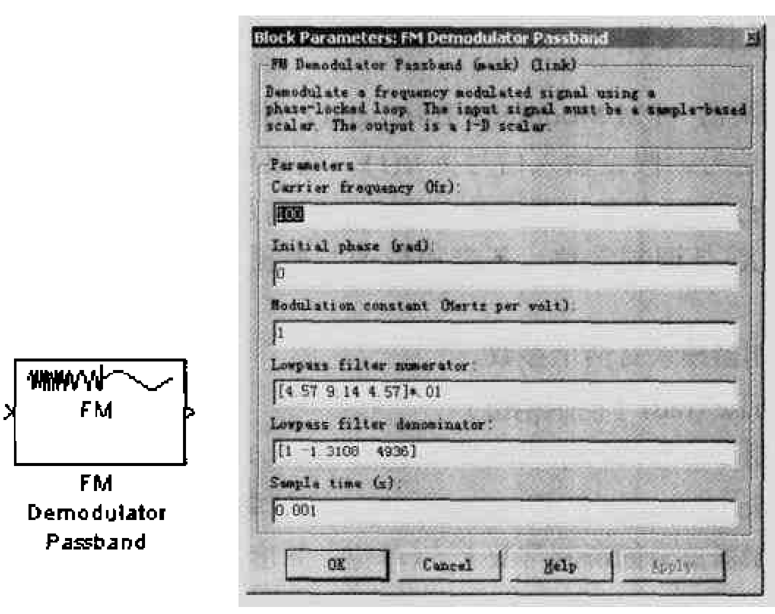



图 8-16　基带频率解调器模块及其参数设置对话框

频带频率解调器使用了一个低通滤波器，这个低通滤波器由 Lowpass filter numerator 和 Lowpass filter denominator 两个参数确定。基带频率解调器主要有以下几个参数。

■ Carrier frequency (Hz)（载波频率）

频带频率调制信号的载波频率（单位：Hz）。

■ Initial phase (rad)（初始相位）

频带频率调制信号的初始相位 $\theta$（单位：弧度）。

■ Modulation constant (Hertz per volt)（调制常数）

频带频率调制信号的调制常数 $K_c$（单位：Hz/伏特）。

■ Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

■ Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

■ Sample time（抽样间隔）

频带频率解调器输出信号的抽样间隔（单位：秒）。

# 8.3 模拟相位调制

相位调制 PM（Phase Modulation）产生的输出信号的载波相位与输入信号的幅度成正比，因此，从广义上说，频率调制是一种特殊的相位调制方式。假设输入信号为 $u(t)$，则输出信号 $y(t)=\exp(j\theta+jK_c u(t))$，其中 $\theta$ 是初始相位，$K_c$ 是调制常数。本节介绍基带相位调制的原理和使用方法。

## 8.3.1 基带相位调制

基带相位调制器（PM Modulator Baseband）对输入的实信号实施相位调制，产生的复数形式的输出信号。假设输入信号为 $u(t)$，输出信号为 $y(t)$，则输出信号 $y(t)$ 的频率随着输入信号 $u(t)$ 的幅度变化而变化，两者之间满足关系式 $y(t)=\exp(j\theta+jK_c u(t))$，其中 $\theta$ 是初始相位，$K_c$ 是调制常数。基带相位调制器模块及其参数设置对话框如图 8-17 所示。

基带相位调制器模块有两个参数。

■ Initial phase (rad)（初始相位）

基带相位调制器信号的初始相位 $\theta$（单位：弧度）。

■ Modulation constant (Radians per volt)（调制常数）

基带相位调制器信号的调制常数 $K_c$（单位：弧度/伏特）。

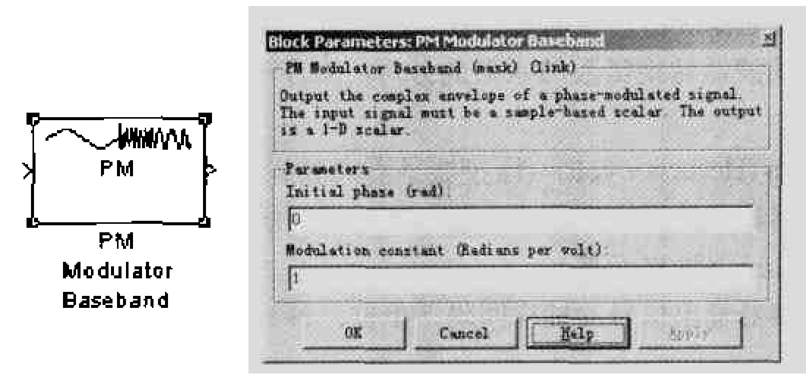



图 8-17　基带相位调制器模块及其参数设置对话框

基带相位解调器（PM Demodulator Baseband）从输入的基带相位调制信号中解调出原始的信息序列。基带相位解调器的输入信号是抽样形式的复信号，输出信号是实信号。图 8-18 所示是基带相位解调器模块及其参数设置对话框。

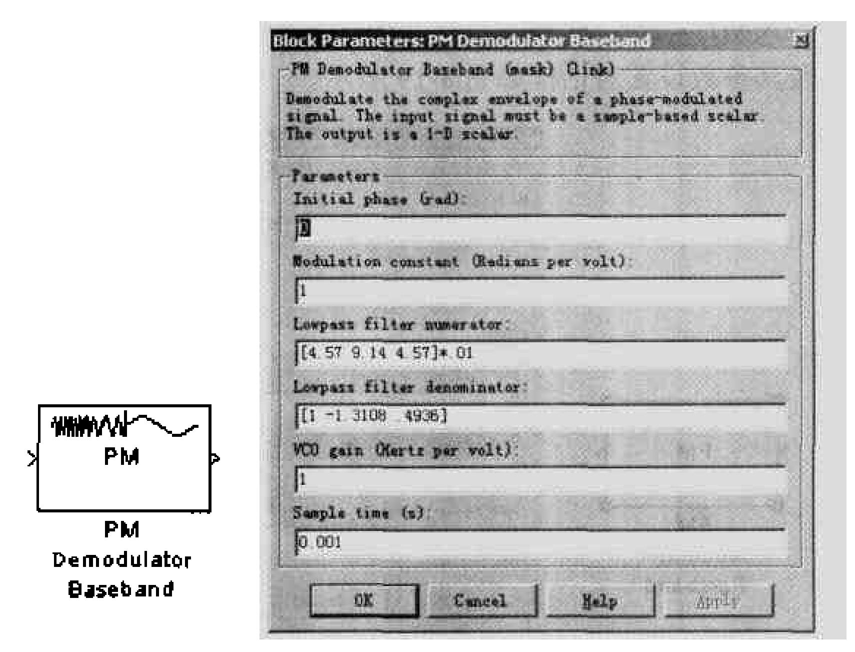



图 8-18　基带相位解调器模块及其参数设置对话框

基带相位解调器使用了一个压控振荡器 VCO（Voltage-Controlled Oscillator）和一个低通滤波器，其中压控振荡器的灵敏度由参数 VCO Gain 确定，低通滤波器的转换多项式由 Lowpass filter numerator 和 Lowpass filter denominator 两个参数确定。基带相位解调器主要有以下几个参数。

■　Initial phase (rad)（初始相位）

与基带相位解调器相对应的调制器的初始相位 $\theta$（单位：弧度）。

■　Modulation constant (Radians per volt)（调制常数）

与基带相位解调器相对应的调制器的调制常数 $K_c$（单位：弧度/伏特）。

■　Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

■ Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

■ VCO gain (Hertz per volt)（压控振荡器增益）

基带相位解调器中的压控振荡器对输入信号的灵敏度（单位：Hz/伏特）。

■ Sample time（抽样间隔）

基带相位解调器输出信号的抽样间隔（单位：秒）。

## 8.3.2 频带相位调制

频带相位调制器（PM Modulator Passband）对输入的实信号实施相位调制。假设输入信号为$u(t)$，输出信号为$y(t)$，则输出信号$y(t)$的频率随着输入信号$u(t)$的幅度变化而变化，两者之间满足关系式$y(t)=\cos(2\pi f_c t+K_c u(t)+\theta)$，其中$\theta$是初始相位，$K_c$是调制常数。频带相位调制器模块及其参数设置对话框如图 8-19 所示。

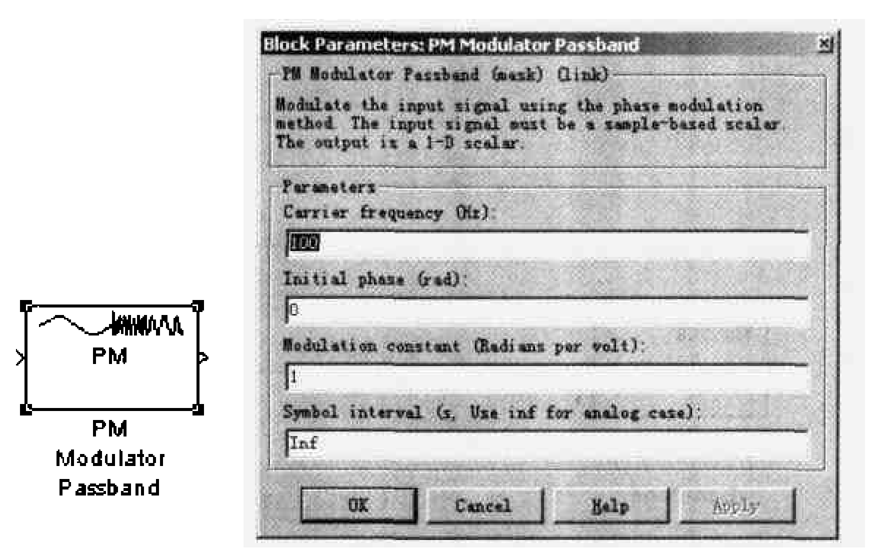



图 8-19 频带相位调制器模块及其参数设置对话框

频带相位调制器模块有如下几个参数。

■ Carrier frequency (Hz)（载波频率）

频带相位调制信号的载波频率（单位：Hz）。

■ Initial phase (rad)（初始相位）

频带相位调制器信号的初始相位$\theta$（单位：弧度）。

■ Modulation constant (Radians per volt)（调制常数）

频带相位调制器信号的调制常数$K_c$（单位：弧度/伏特）。

■ Symbol interval (s)（符号间隔）

频带相位调制器传输一个信息位的周期（单位：秒），缺省值是 Inf。

频带相位解调器（PM Demodulator Passband）从输入的频带相位调制信号中解调出原始的信息序列，它的输入信号和输出信号都是实数形式的抽样信号。图 8-20 所示是频带相位解调器模块及其参数设置对话框。

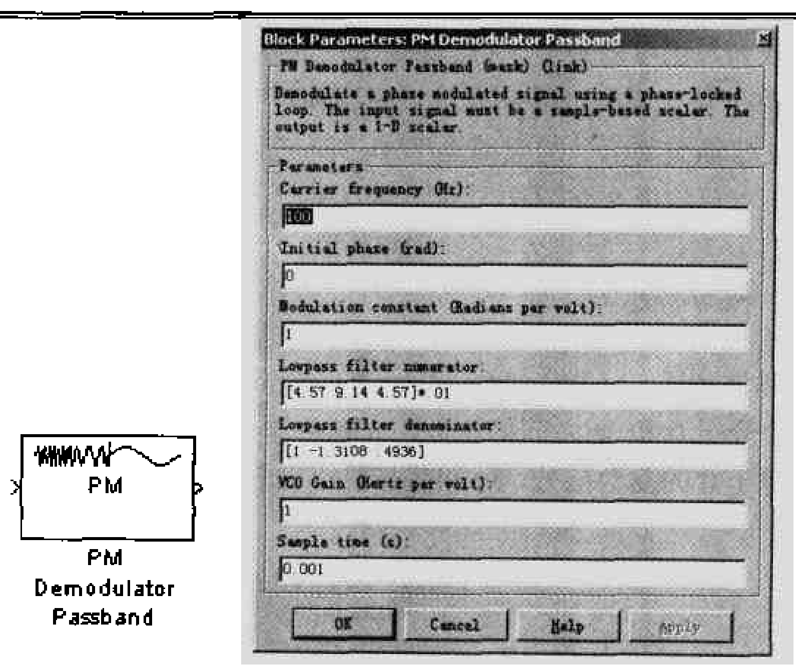



图 8-20　频带相位解调器模块及其参数设置对话框

频带相位解调器中的压控振荡器的灵敏度由参数 VCO Gain 确定，低通滤波器的转换多项式由 Lowpass filter numerator 和 Lowpass filter denominator 两个参数确定。频带相位解调器主要有以下几个参数。

■　Carrier frequency (Hz)（载波频率）

频带相位调制信号的载波频率（单位：Hz）。

■　Initial phase (rad)（初始相位）

与基带相位解调器相对应的调制器的初始相位 $\theta$（单位：弧度）。

■　Modulation constant (Radians per volt)（调制常数）

与基带相位解调器相对应的调制器的调制常数 $K_C$（单位：弧度/伏特）。

■　Lowpass filter numerator（低通滤波器分子）

本参数表示低通滤波器转换函数的分子，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分子。

■　Lowpass filter denominator（低通滤波器分母）

本参数表示低通滤波器转换函数的分母，它是一个向量，向量中的元素以指数从高到低的顺序表示低通滤波器转换函数的系数的分母。对于有限冲击响应（FIR）滤波器，本参数设置为 1。

■　VCO gain (Hertz per volt)（压控振荡器增益）

基带相位解调器中的压控振荡器对输入信号的灵敏度（单位：Hz/伏特）。

■　Sample time（抽样间隔）

基带相位解调器输出信号的抽样间隔（单位：秒）。

## 8.4　数字幅度调制

与模拟信号的幅度调制类似地，对于数字方式的幅度调制信号，它的幅度也随着输入的输入的数字信号的变化而相应地发生变化。M 相数字幅度调制的输入信号既可以是一个介于

0 和 $M$ 之间的整数 $m$，也可以是一个特定长度的二进制序列 $b_n b_{n-1} \cdots b_1$，其中序列的长度满足 $n = \lceil \log_2 M \rceil$。数字幅度调制可以分成脉幅调制和正交幅度调制，它们产生的基带调制信号都是复信号，但是脉幅调制信号的虚部等于 0，而正交幅度调制的实部和虚部都随着输入信号的变化而变化。本节将依次介绍这几种调制方式。

## 8.4.1 基带脉幅调制

M 相基带脉幅调制器（M-PAM Modulator Baseband）对输入信号实施 M 相脉幅调制，它的输入信号是一个介于 0 和 $M$ 之间的整数，或者是一个长度等于 $\lceil \log_2 M \rceil$ 的二进制序列，输出信号则是虚部等于 0 的复数信号。M 相基带脉幅调制器模块及其参数设置对话框如图 8-21 所示。

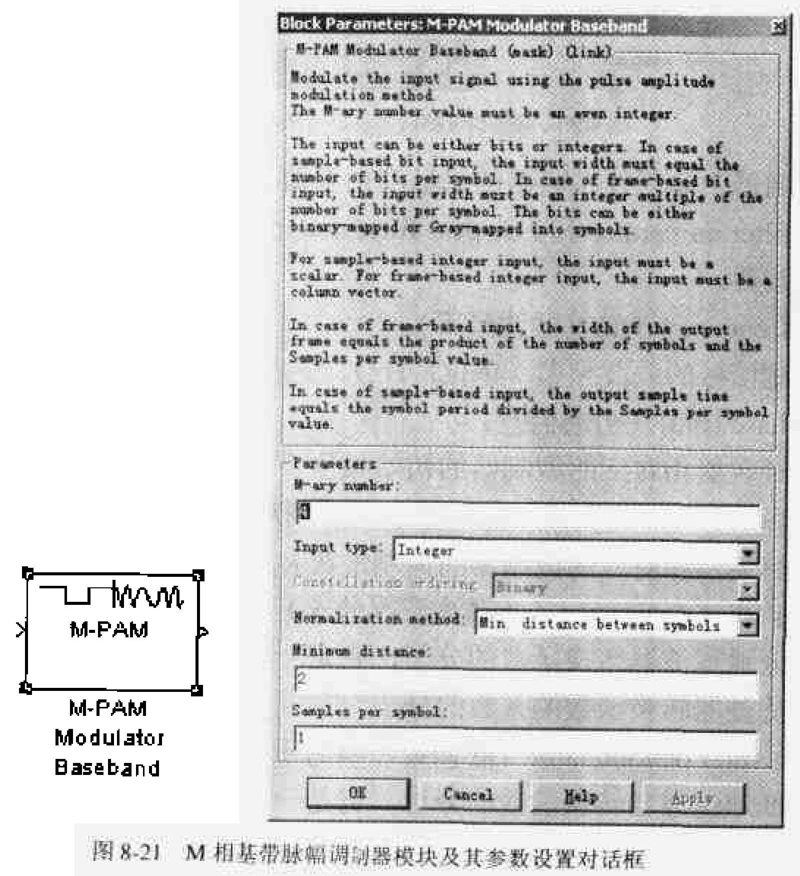



图 8-21　M 相基带脉幅调制器模块及其参数设置对话框

M 相基带脉幅调制器模块主要有以下几个参数。

- **M-ary number（M 相数）**

M 相基带脉幅调制的相位数 M，即 M 相基带脉幅调制星座图中点的数目，它应该设置为一个偶数。

- **Input type（输入信号类型）**

M 相基带脉幅调制输入信号的类型，它既可以是整数（Integer），也可以是二进制数（Bit）。如果输入信号类型设置为 Bit，则 M-ary number 应该等于 $2^k, k > 0$。

- **Constellation ordering（星座图编码方式）**

当输入信号类型设置为 Bit 时，Constellation ordering 用来指定输入的每 $k$ 个二进制符号

与 M 相基带脉幅调制星座图中各个点的对应关系：如果星座图编码方式设置为 Binary，MATLAB 把输入的 $k$ 个二进制符号当作一个二进制序列；如果星座图编码方式设置为 Gray，则 MATLAB 把输入的 $k$ 个二进制符号当作一个 Gray 码。

Gray 码是与自然二进制序列不同的一种编码，并且任意两个相邻的 Gray 码之间的距离都等于 1。Gray 码与二进制序列之间的换算关系如下：假设二进制序列等于 $b_n b_{n-1} \cdots b_1$，则与之相对应的 Gray 码序列 $g_n g_{n-1} \cdots g_1 = b_n b_{n-1} \cdots b_1 \oplus b_{n-1} b_{n-2} \cdots b_2$，其中 $\oplus$ 表示模二加法。下面的程序段用于产生一个与二进制序列相对应的八进制的 Gray 码。例如，Gray 码中的 111 等价于二进制序列的 101。

```
>> M=8;
>> m=[0:M-1]';
>> de2bi(bitxor(m,floor(m/2)), log2(M),'left-msb')
ans =
     0     0     0
     0     0     1
     0     1     1
     0     1     0
     1     1     0
     1     1     1
     1     0     1
     1     0     0
```

■ Normalization method（归一化方法）

M 相基带脉幅调制信号星座图的缩放方式：当选择 Min. distance between symbols 时，星座图中距离最近的两个点之间的距离由参数 Minimum distance 确定；当选择 Average Power 时，星座图中所有点的平均功率由参数 Average power 确定；当选择 Peak Power 时，星座图中各个点的最大功率由参数 Peak power 确定。

■ Minimum distance（星座图的最小距离）

当 Normalization method 设置为 Min. distance between symbols 时，Minimum distance 表示星座图中距离最近的两个点之间的距离。

■ Average power (watts)（星座图的平均功率）

当 Normalization method 设置为 Average Power 时，Average power 表示星座图中所有点的平均功率。

■ Peak power (watts)（星座图的最大功率）

当 Normalization method 设置为 Peak Power 时，Peak power 表示星座图中各个点的最大功率。

■ Samples per symbol（抽样次数）

M 相基带脉幅调制器输出信号的抽样频率可以高于输入信号的抽样频率，本参数表示对应于每个输入符号产生的输出信号的个数，它应该是一个正整数。

图 8-22 所示是一个四相基带脉幅调制的输出信号波形（示波器中只示出了调制信号的实部，调制信号的虚部等于 0）。在这个四相基带脉幅调制器中，输入信号是介于 0 和 3

之间的整数，抽样周期为 0.01 秒，星座图中各点之间的最小距离设置为 2。从图 8-22 所示中可以看到，输出的基带调制信号的取值是-3、-1、1 和 3，分别对应于输入信号 0、1、2、3。

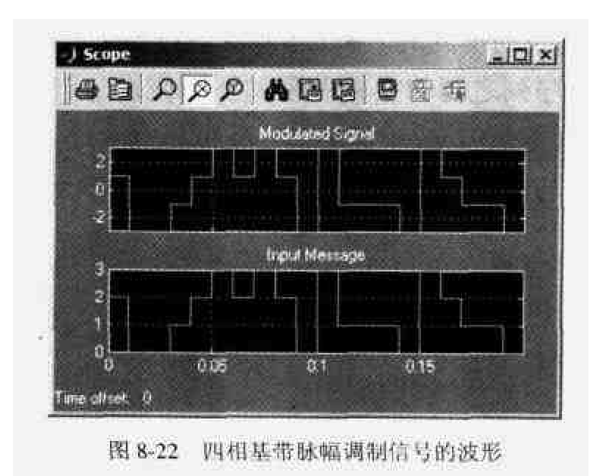



图 8-22　四相基带脉幅调制信号的波形

M 相基带脉幅解调器（M-PAM Demodulator Baseband）对基带脉冲幅度调制信号进行解调，其模块和参数设置对话框如图 8-23 所示。

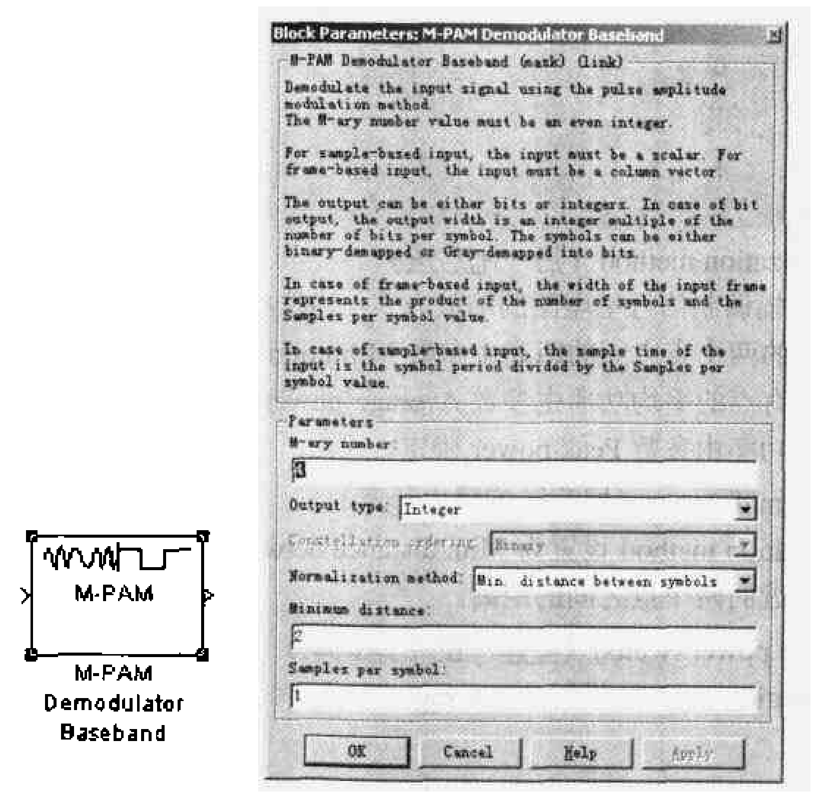



图 8-23　M 相基带脉幅解调器模块和参数设置对话框

M 相基带脉幅解调器的参数与 M 相基带脉幅调制器相同。

## 8.4.2　频带脉幅调制

M 相频带脉幅调制器（M-PAM Modulator Passband）对输入信号实施 M 相脉幅调制，输出载波频率等于 $f_c$ 的频带调制信号。图 8-24 所示是 M 相频带脉幅调制器模块及其参数设置对话框。

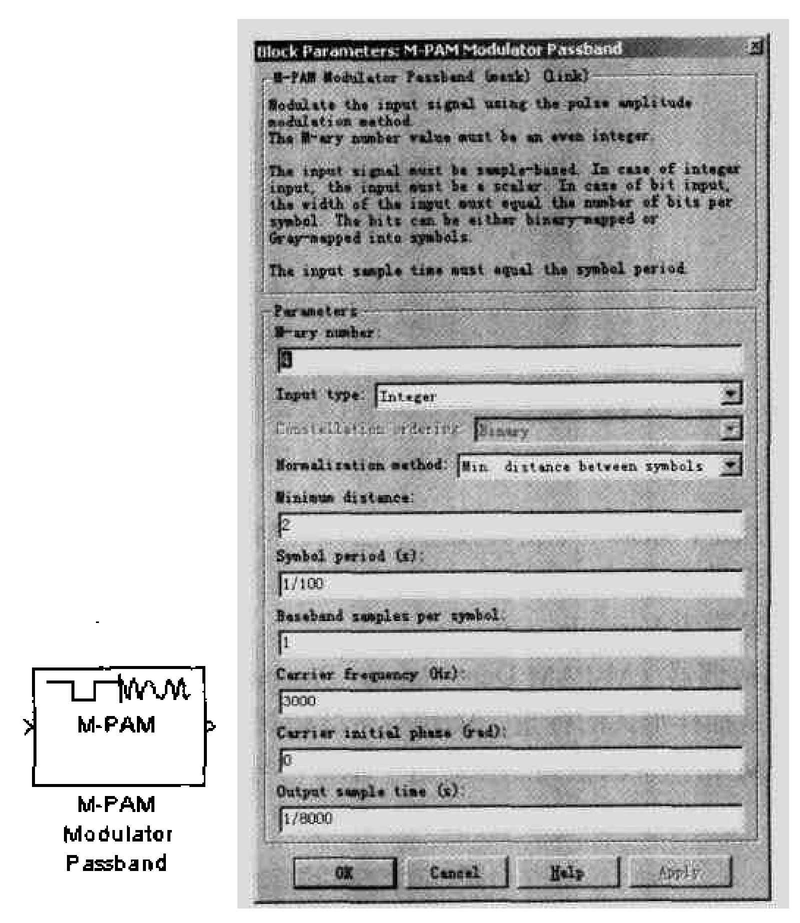



图 8-24 M 相频带脉幅调制器模块及其参数设置对话框

M 相频带脉幅调制器模块中的参数 M-ary number、Input type、Constellation ordering、Normalization method、Minimum distance、Average power 以及 Peak power 都与 M 相基带脉幅调制器模块中的同名参数相同。除此之外，M 相频带脉幅调制器模块还有以下几个参数。

■ Symbol period (s)（输入符号间隔）

M 相频带脉幅调制器输入信号的抽样间隔（单位：秒）。

■ Baseband samples per symbol（基带符号抽样数）

M 相频带脉幅调制器每个输入符号对应的抽样个数。

■ Carrier frequency (Hz)（载波频率）

M 相频带脉幅调制信号的载波频率（单位：Hz）。

■ Carrier initial phase (rad)（载波初始相位）

M 相频带脉幅调制信号使用的载波的初始相位（单位：弧度）。

■ Output sample time（输出信号抽样间隔）

M 相频带脉幅调制器输出信号的抽样间隔（单位：秒）。

图 8-25 所示是一个四相频带脉幅调制器的输出信号，其中载波频率等于 300Hz，输入信号的抽样周期等于 0.01 秒，输出信号的抽样周期等于 1/8000 秒，星座图中各点之间的最小距离等于 2。从图中可以看到，频带脉幅调制信号除了幅度的变化外，还存在着载波相位的变化。例如，当输入信号从 1 变成 2 时，频带脉幅调制信号的幅度保持不变，但是相位翻转 180 度。

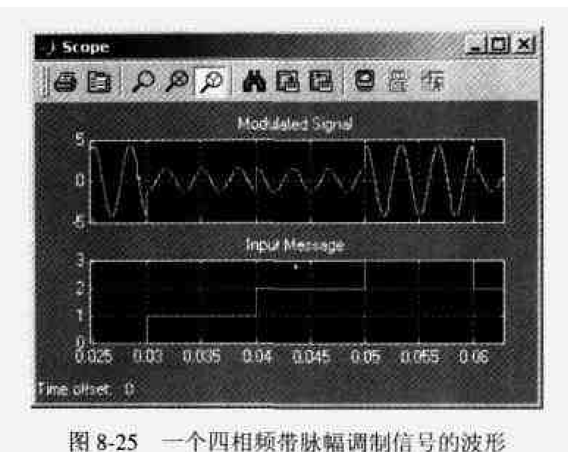



图 8-25 一个四相频带脉幅调制信号的波形

需要注意的一点是，如果一个仿真模型中包含了 M 相频带脉幅调制器模块，则这个仿真模型必须使用可变步长（variable-step solver）进行仿真，即把仿真参数 Simulation | Simulation parameters | Solver 设置为 Variable-step。

M 相频带脉幅解调器（M-PAM Demodulator Passband）对基带脉冲幅度调制信号进行解调，它的输入信号是抽样形式的标量。M 相频带脉幅调制器首先把频带信号转换成相应的基带信号，然后用过 M 相基带脉幅调制器对这个基带信号进行解调，其模块和参数设置对话框如图 8-26 所示。

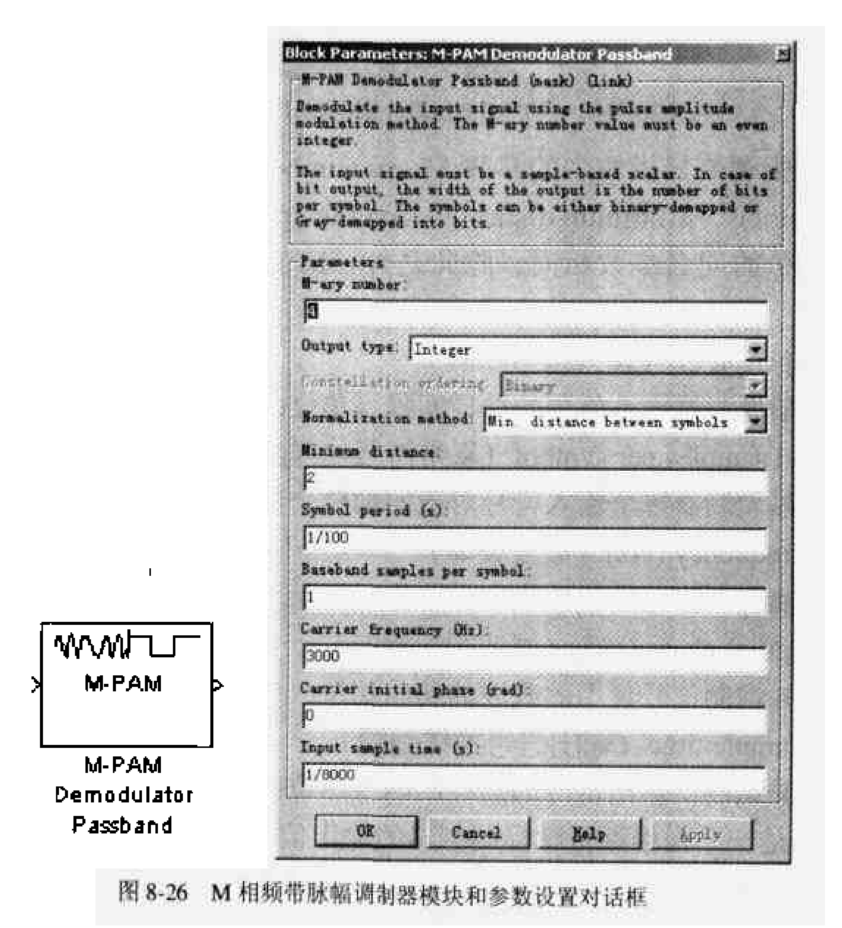



图 8-26 M 相频带脉幅调制器模块和参数设置对话框

M 相频带脉幅解调器的参数与 M 相频带脉幅调制器相同。

## 8.4.3　基带正交幅度调制

基带正交幅度调制器（General QAM Modulator Baseband）的星座图中有 $M$ 个点，输入信号是介于 0 和 $M$-1 之间的标量或帧结构的列向量，并且把输入信号 $m$ 映射为星座图中的第 $m$+1 个点。基带正交幅度调制器模块及其参数设置对话框如图 8-27 所示。

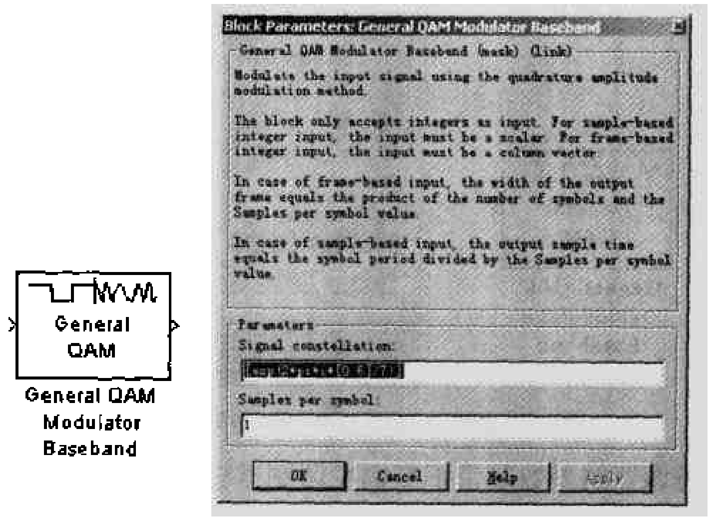



图 8-27　基带正交幅度调制器模块及其参数设置对话框

基带正交幅度调制器模块主要有两个参数。

- Signal constellation（信号星座图）

信号星座图是一个向量，向量中的元素既可以是实数，也可以是复数，它们对应于星座图中的点的坐标。

- Samples per symbol（符号抽样数）

基带正交幅度调制器的输出信号的频率可以高于输入信号的频率，本参数表示对应于每个输入符号产生的输出采样点的个数。

图 8-28 所示是一个四相基带正交幅度调制信号的波形图和发散图，其中四相基带正交幅度调制中的参数 Signal constellation 设置为[1+$i$ 1-$i$ -1+$i$ -1-$i$]。从输出信号的波形图中可以看到，星座图中的各个点分别对应于输入信号等于 0、1、2、3 时的输出。这个四相基带正交幅度调制信号的发散图包含了 4 个点，它们分别等于星座图中的 4 个点。

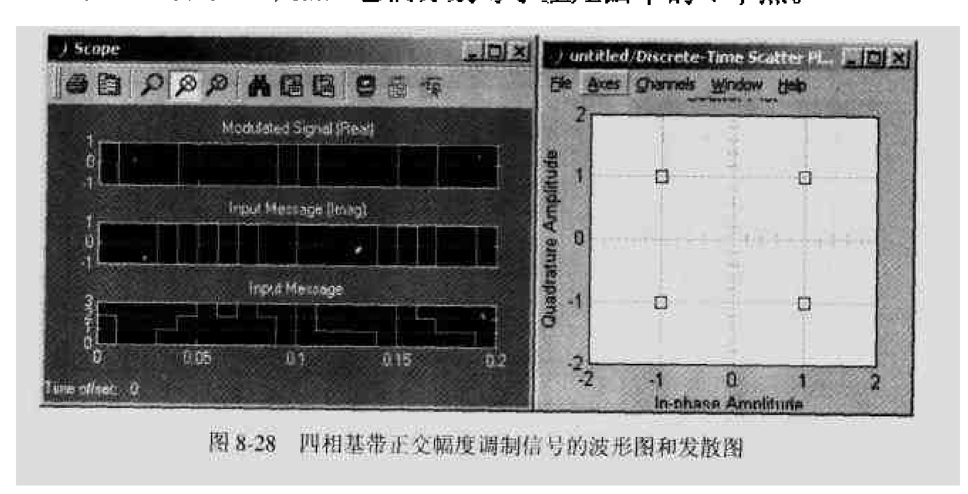



图 8-28　四相基带正交幅度调制信号的波形图和发散图

基带正交幅度解调器（General QAM Demodulator Baseband）的输入信号是正交幅度调制基带复信号，它可以是一个标量，也可以是帧结构的列向量。基带正交幅度解调器根据星座图把输入信号 $m$ 映射为整数 $m$-1，其模块及参数设置对话框如图 8-29 所示。

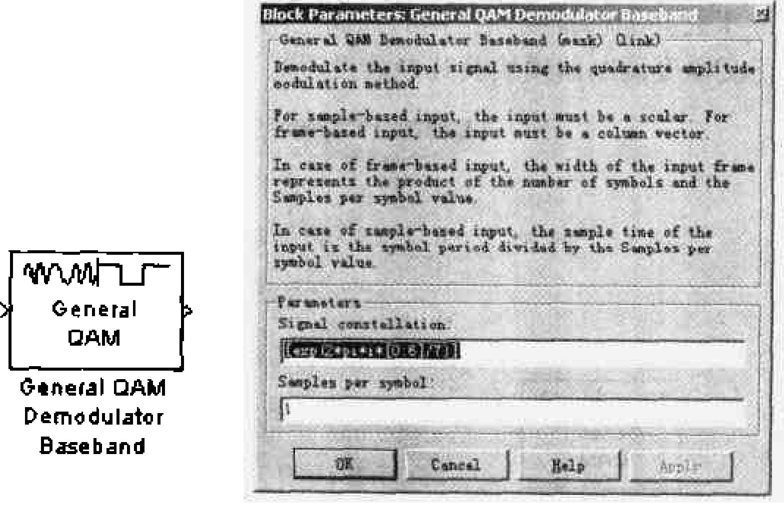



图 8-29　基带正交幅度解调器模块及参数设置对话框

基带正交幅度解调器的参数基带正交幅度调制器相同。

### 8.4.4　频带正交幅度调制

频带正交幅度调制器（General QAM Modulator Passband）的星座图中有 $M$ 个点，它的输入信号是介于 0 和 $M$-1 之间的标量 $m$，并且 $m$ 映射为星座图中的第 $m$+1 个点。频带正交幅度调制器模块及其参数设置对话框如图 8-30 所示。

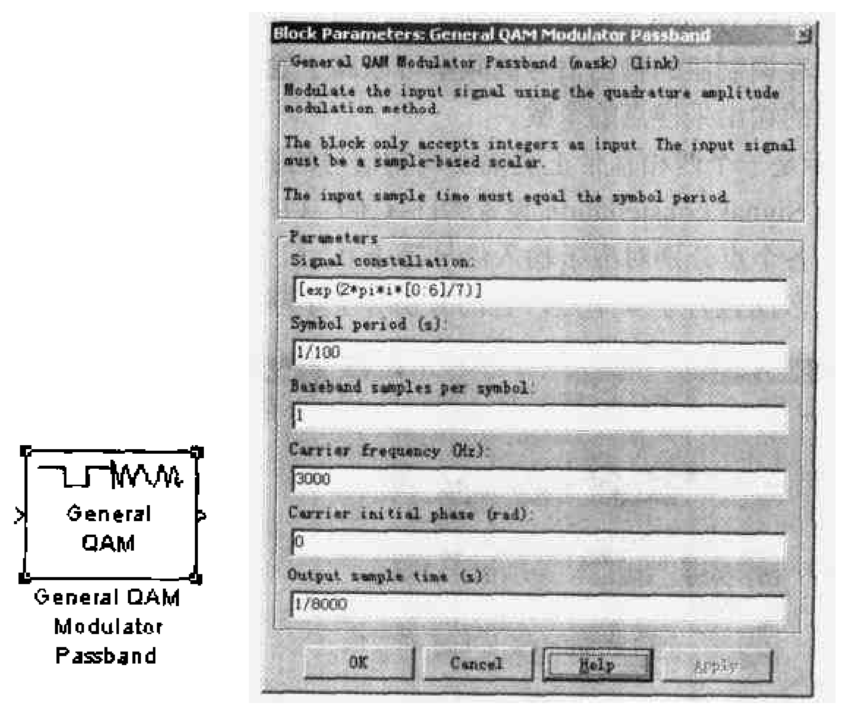



图 8-30　频带正交幅度调制器模块及其参数设置对话框

频带正交幅度调制器模块有以下几个参数。

■　Signal constellation（信号星座图）

信号星座图是一个向量，向量中的元素既可以是实数，也可以是复数，它们对应于星座图中的点的坐标。

■　Symbol period (s)（符号周期）

频带正交幅度调制器的输入信号的抽样周期。

■　Baseband samples per symbol（每个基带符号的抽样数）

频带正交幅度调制器每个输入的基带符号的抽样个数。

■　Carrier frequency (Hz)（载波频率）

频带正交幅度调制器载波的频率（单位：Hz）。

■　Carrier initial phase (rad)（载波初始相位）

频带正交幅度调制器载波的初始相位（单位：弧度）。

■　Output sample time（输出信号抽样间隔）

频带正交幅度调制器输出信号的抽样间隔（单位：秒）。

频带正交幅度解调器（General QAM Demodulator Passband）的输入信号是复数形式的标量，它根据星座图把输入信号 $m$ 映射为整数 $m$-1。图 8-31 所示是频带正交幅度解调器模块及参数设置对话框。

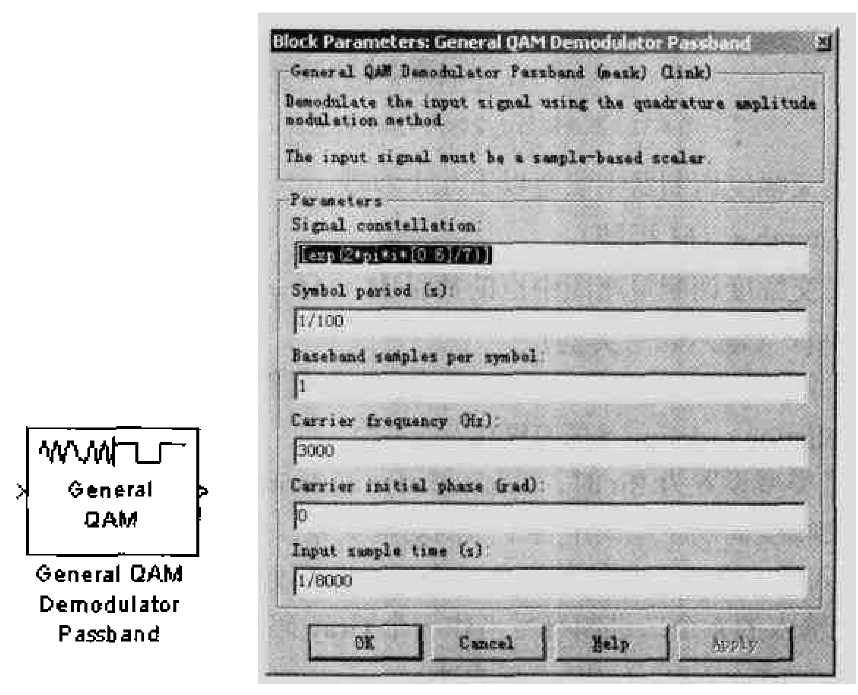



图 8-31　频带正交幅度解调器模块及参数设置对话框

频带正交幅度解调器模块的参数频带正交幅度调制器模块相同。

## 8.4.5　基带矩形正交幅度调制

基带矩形正交幅度调制器（Rectangular QAM Modulator Baseband）通过矩形结构的星座图对输入信号实施 M 相正交幅度调制，产生基带调制信号，其模块框图和参数设置对话框如图 8-32 所示。

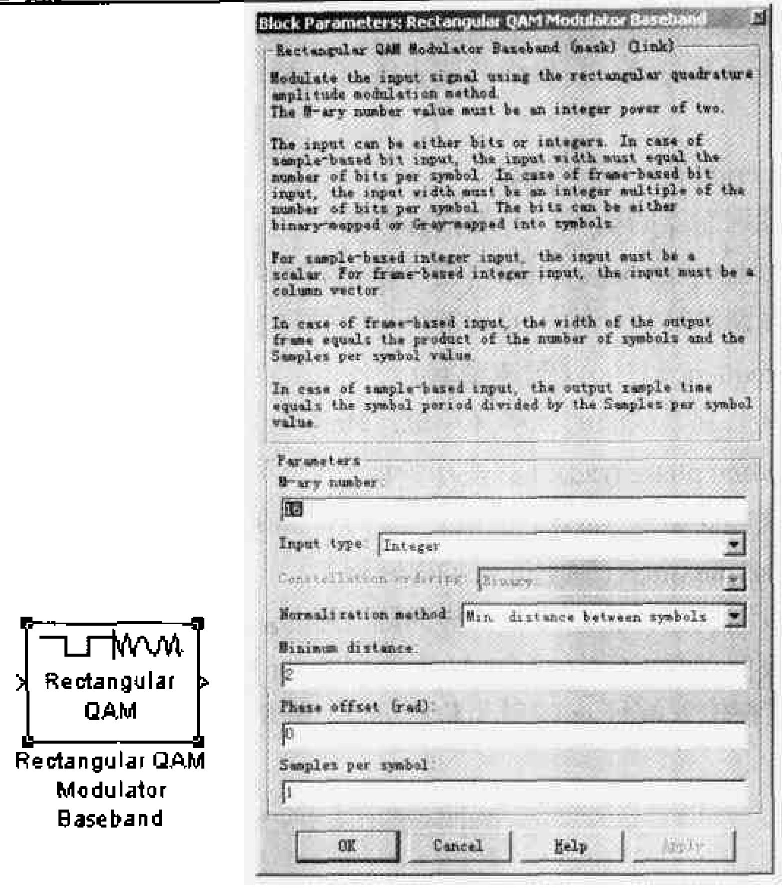



图 8-32　基带矩形正交幅度调制器模块及其参数设置对话框

基带矩形正交幅度调制器主要有以下几个参数。

■　M-ary number（M 相数）

基带矩形正交幅度调制星座图中点的数目 $M$，$M = 2^k, k > 0$。

■　Input type（输入信号类型）

基带矩形正交幅度调制输入信号的类型，它既可以是整数（Integer），也可以是二进制数（Bit）。

■　Constellation ordering（星座图编号方式）

当输入信号类型设置为 Bit 时，Constellation ordering 用来指定输入的每 $k$ 个二进制符号与基带矩形正交幅度调制星座图中各个点的对应关系：如果星座图编号方式设置为 Binary，MATLAB 把输入的 $k$ 个二进制符号当作一个二进制序列；如果星座图编号方式设置为 Gray，则 MATLAB 把输入的 $k$ 个二进制符号当作一个 Gray 码。

■　Normalization method（归一化方法）

基带矩形正交幅度调制信号星座图的缩放方式：当选择 Min. distance between symbols 时，星座图中距离最近的两个点之间的距离由参数 Minimum distance 确定；当选择 Average Power 时，星座图中所有点的平均功率由参数 Average power 确定；当选择 Peak Power 时，星座图中各个点的最大功率由参数 Peak power 确定。

■　Minimum distance（星座图的最小距离）

当 Normalization method 设置为 Min. distance between symbols 时，Minimum distance 表示星座图中距离最近的两个点之间的距离。

■　Average power (watts)（星座图的平均功率）

当 Normalization method 设置为 Average Power 时，Average power 表示星座图中所有点的平均功率。

■　Peak power (watts)（星座图的最大功率）

当 Normalization method 设置为 Peak Power 时，Peak power 表示星座图中各个点的最大功率。

■　Phase offset (rad)（相位偏移）

基带矩形正交幅度调制器的星座图的旋转角度（单位：弧度）。

■　Samples per symbol（抽样次数）

基带矩形正交幅度调制器对应于每个输入符号产生的输出信号的抽样点的个数。

基带矩形正交幅度解调器（Rectangular QAM Demodulator Baseband）对基带正交幅度调制信号进行解调，它使用的星座图与相应的基带矩形正交幅度调制器相同，其模块框图和参数设置对话框如图 8-33 所示。

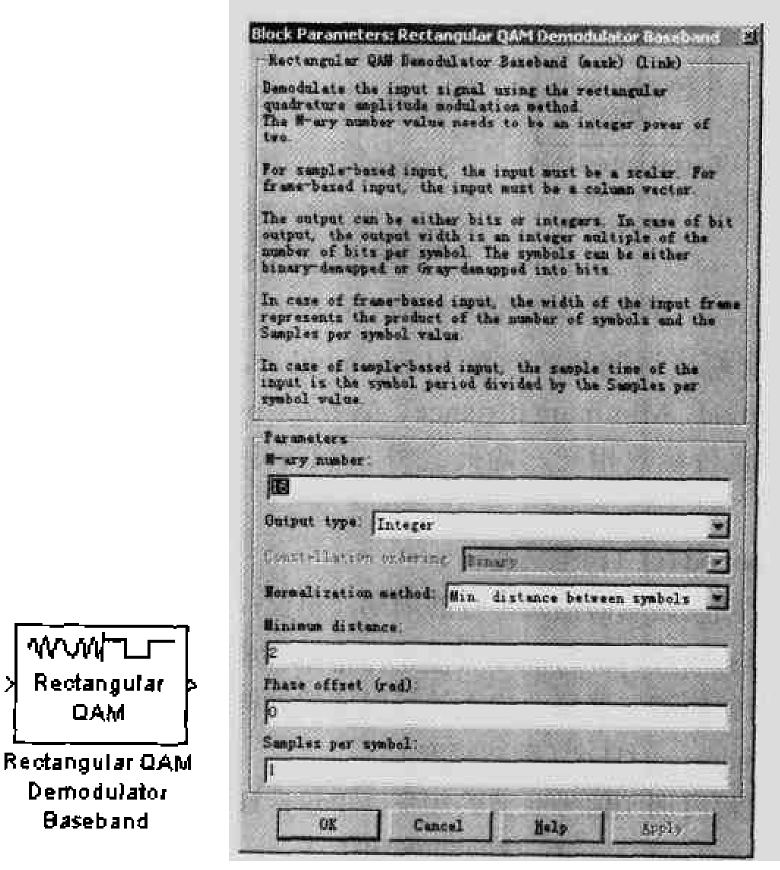



图 8-33　基带矩形正交幅度解调器模块及其参数设置对话框

基带矩形正交幅度解调器的参数与基带矩形正交幅度调制器相同。

## 8.4.6　频带矩形正交幅度调制

频带矩形正交幅度调制器（Rectangular QAM Modulator Baseband）通过矩形结构的星座图对输入信号实施 M 相正交幅度调制，产生频带调制信号。它使用基带矩形正交幅度调制器对输入信号进行基带调制，然后把这个基带信号转换成频带信号。图 8-34 所示是频带矩形正交幅度调制器模块及其参数设置对话框。

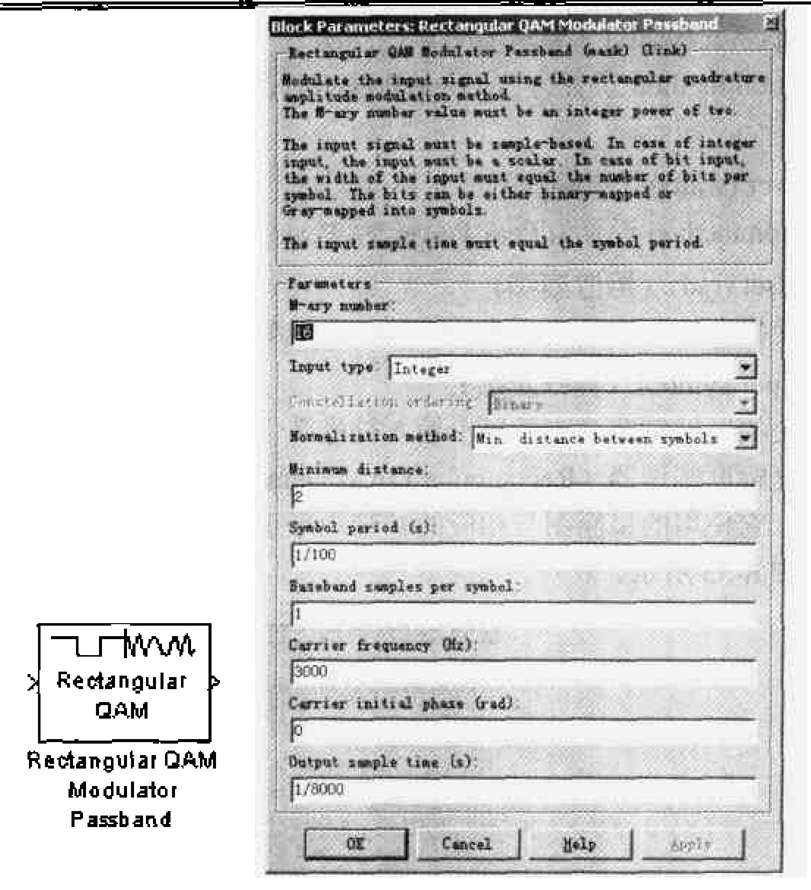



图 8-34 频带矩形正交幅度调制器模块及其参数设置对话框

频带矩形正交幅度调制器的参数 M-ary number、Input type、Constellation ordering、Normalization method、Minimum distance、Average power 以及 Peak power 与基带矩形正交幅度调制器的同名参数相同。除此之外，频带矩形正交幅度调制器还有以下几个参数。

■ Symbol period (s)（符号周期）

频带矩形正交幅度调制器输入符号的周期（单位：秒）。

■ Baseband samples per symbol（基带符号的抽样数）

频带矩形正交幅度调制器每个输入符号对应的抽样点的个数。

■ Carrier frequency (Hz)（载波频率）

频带矩形正交幅度调制信号载波的频率（单位：Hz）。

■ Carrier initial phase (rad)（载波初始相位）

频带矩形正交幅度调制信号载波的初始相位（单位：弧度）。

■ Output sample time（输出信号抽样间隔）

频带矩形正交幅度调制器输出的频带信号的抽样间隔（单位：秒）。

频带矩形正交幅度解调器（Rectangular QAM Demodulator Passband）的输入信号是抽样形式的标量，它首先把频带信号转换成基带信号，然后通过 M 相基带脉幅解调器对这个基带信号进行解调，其模块框图和参数设置对话框如图 8-35 所示。

频带矩形正交幅度解调器的参数与频带矩形正交幅度调制器相同。

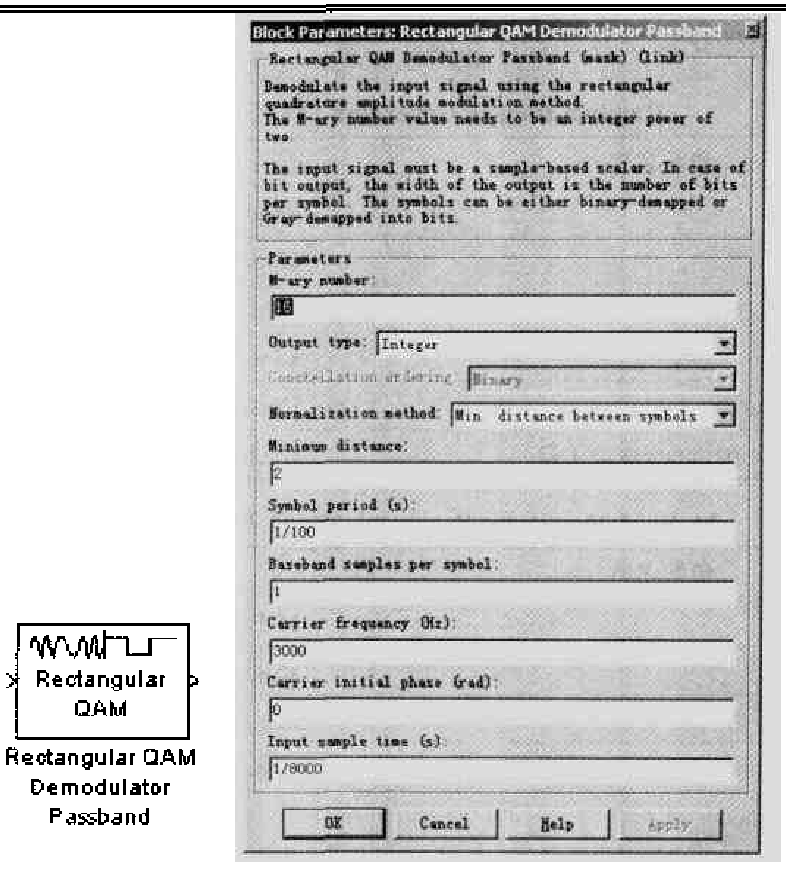



图 8-35　频带矩形正交幅度解调器模块及其参数设置对话框

## 8.4.7　实例 8.1——数字幅度调制的抗噪声性能

前面我们介绍了 3 种数字幅度调制方式，包括脉幅调制器（PAM）和正交幅度调制器（QAM）。本节我们将用一个实例程序来考察这两种调制方式的抗噪声性能。我们将创建两个仿真模型，分别用于实现脉幅调制和正交幅度调制。图 8-36 所示是对信号实施脉幅调制的仿真模型。

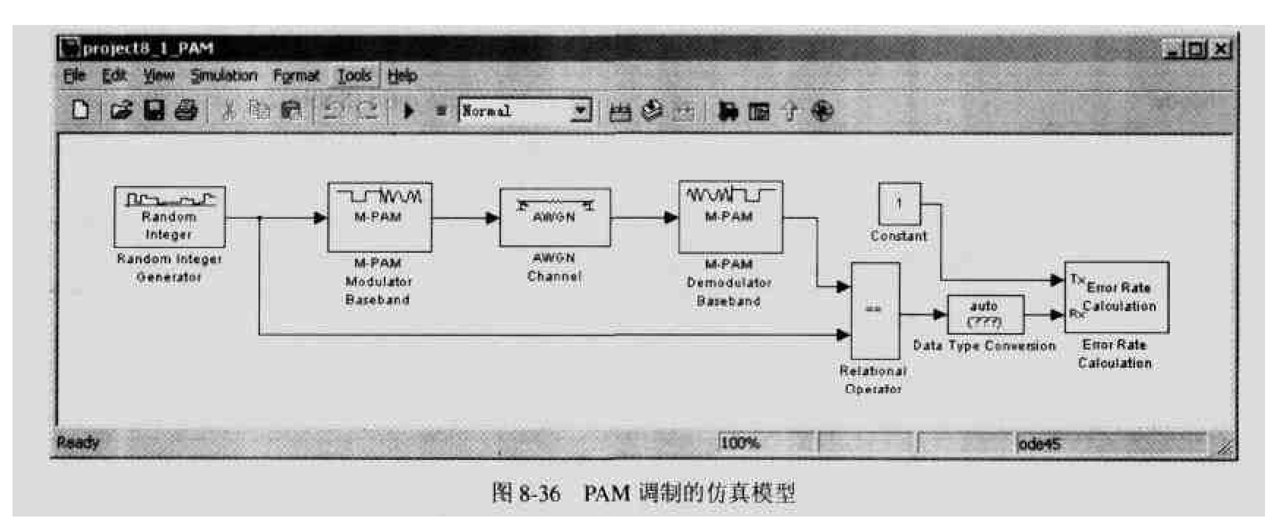



图 8-36　PAM 调制的仿真模型

在 PAM 调制的仿真模型中，Random Integer Generator（随机整数产生器）产生一个八进制整数序列，这个整数序列通过 M-PAM Modulator Baseband（PAM 基带调制器模块）进行调制，得到基带调制信号。表 8-1 和表 8-2 列出了随机整数产生器和 PAM 基带调制器模块的参数设置情况，其中 xSignalLevel、xInitialSeed 和 xSampleTime 分别表示整数序列的相数、随

机整数产生器的初始化种子及其抽样间隔。

表 8-1　　Random Integer Generator（随机整数产生器）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Random Integer Generator |
| M-ary number | xSignalLevel |
| Initial seed | xInitialSeed |
| Sample time | xSampleTime |
| Frame-based outputs | Unchecked |
| Interlpret vector parameters as 1-D | Unchecked |

表 8-2　　M-PAM Modulator Baseband（PAM 基带调制器模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | M-PAM Modulator Baseband |
| M-ary number | xSignalLevel |
| Input type | Integer |
| Normalization method | Min. distance between symbols |
| Minimum diatance | 2 |
| Samples per symbol | 1 |

PAM 基带调制器模块产生的基带调制信号经过 AWGN Channel（加性高斯白噪声信道）后叠加了一定强度的噪声，这个信号由 M-PAM Demodulator Baseband（PAM 基带解调器模块）进行解调。加性高斯白噪声信道模块和 PAM 基带解调器模块的参数设置如表 8-3 和表 8-4 所示。

表 8-3　　AWGN Channel（加性高斯白噪声模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | AWGN Channel |
| Initial seed | 67 |
| Mode | Signal to noise ratio (SNR) |
| SNR (dB) | xSNR |
| Input signal power (watts) | 1 |

表 8-4　　M-PAM Demodulator Baseband（PAM 基带解调器模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | M-PAM Demodulator Baseband |
| M-ary number | xSignalLevel |
| Input type | Integer |
| Normalization method | Min. distance between symbols |
| Minimum diatance | 2 |
| Samples per symbol | 1 |

最后，Error Rate Calculation（误码率统计模块）对原始信号和解调信号进行比较，统计得到 PAM 调制的误码率，并且把误码信息保存在 MATLAB 工作区变量 xErrorRate 中。误

码率统计模块的参数设置如表 8-5 所示。

表 8-5　**Error Rate Calculation（误码率统计模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Error Rate Calculation |
| Receive delay | 0 |
| Computation delay | 0 |
| Computation mode | Entire frame |
| Output data | Workspace |
| Variable name | xErrorRate |
| Reset port | Unchecked |
| Stop simulation | Unchecked |

到此为止，PAM 仿真模型的设计已介绍完毕。QAM 调制的仿真模型与 PAM 调制模型非常相似，它只是把 PAM 基带调制器模块和解调器模块分别换成 Rectangular QAM Modulator Baseband（QAM 基带调制器模块）和 Rectangular QAM Demodulator Baseband（QAM 基带解调器模块），如图 8-37 所示。

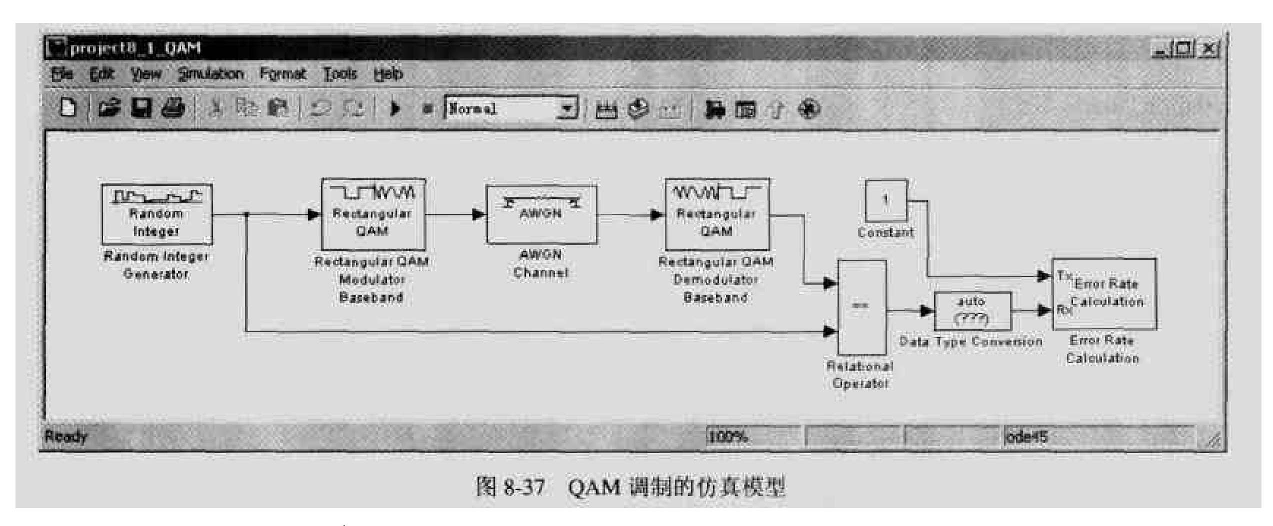



图 8-37　QAM 调制的仿真模型

在 QAM 调制的仿真模型中，信源、信道和信宿都与 PAM 仿真模型保持一致，这样便于在相同条件下比较两种调制方式的性能。表 8-6 和表 8-7 分别列出了 QAM 基带调制器模块和 QAM 基带解调器模块的参数设置情况。

表 8-6　**Rectangular QAM Modulator Baseband（QAM 基带调制器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Rectangular QAM Modulator Baseband |
| M-ary number | xSignalLevel |
| Input type | Integer |
| Normalization method | Min. distance between symbols |
| Minimum diatance | 2 |
| Phase offset (rad) | 0 |
| Samples per symbol | 1 |

表 8-7　　Rectangular QAM Demodulator Baseband（QAM 基带解调器模块）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Rectangular QAM Demodulator Baseband |
| M-ary number | xSignalLevel |
| Input type | Integer |
| Normalization method | Min. distance between symbols |
| Minimum diatance | 2 |
| Phase offset (rad) | 0 |
| Samples per symbol | 1 |

为了比较两种调制方式在不同信噪比条件下的误码性能，需编写了 M 文件 project8_1main.m，用于实现对仿真模型参数的初始化以及循环执行仿真模型。下面的程序段是 project8_1main.m 文件的内容。

```
% 设置调制信号的相数（调制信号是介于 0 和 xSignalLevel-1 之间的整数）
xSignalLevel = 8;
% 设置调制信号的抽样间隔
xSampleTime = 1/100000;
% 设置仿真时间的长度
xSimulationTime = 10;
% 设置随机数产生器的初始化种子
xInitialSeed = 37;

% x 表示信噪比的取值范围
x = 0:10;
% y1 表示 PAM 调制的误符号率
y1 = x;
% y2 表示 QAM 调制的误符号率
y2 = x;

for i = 1:length(x)
    % 信噪比依次取向量 x 的数值
    xSNR = x(i);
    % 执行 PAM 仿真模型
    sim('project8_1_PAM');
    % 从 xErrorRate 中获得调制信号的误符号率
    y1(i) = xErrorRate(1);
end
for i = 1:length(x)
    % 信噪比依次取向量 x 的数值
    xSNR = x(i);
```

```
    % 执行 QAM 仿真模型
    sim('project8_1_QAM');
    % 从 xErrorRate 中获得调制信号的误符号率
    y2(i) = xErrorRate(1);
end

% 绘制信噪比与误符号率的关系曲线
% 蓝色线条表示 PAM 调制，红色线条表示 QAM 调制
semilogy(x,y1,'b',x,y2,'r');
```

仿真结束之后得到如图 8-38 所示的误码率曲线，其中实线表示 PAM 调制的误码率，虚线表示 QAM 调制的误码率。从图 8-38 所示中可以看到，这两种调制方式的误码率是比较相近的，而 PAM 的抗噪声性能略优于 QAM。

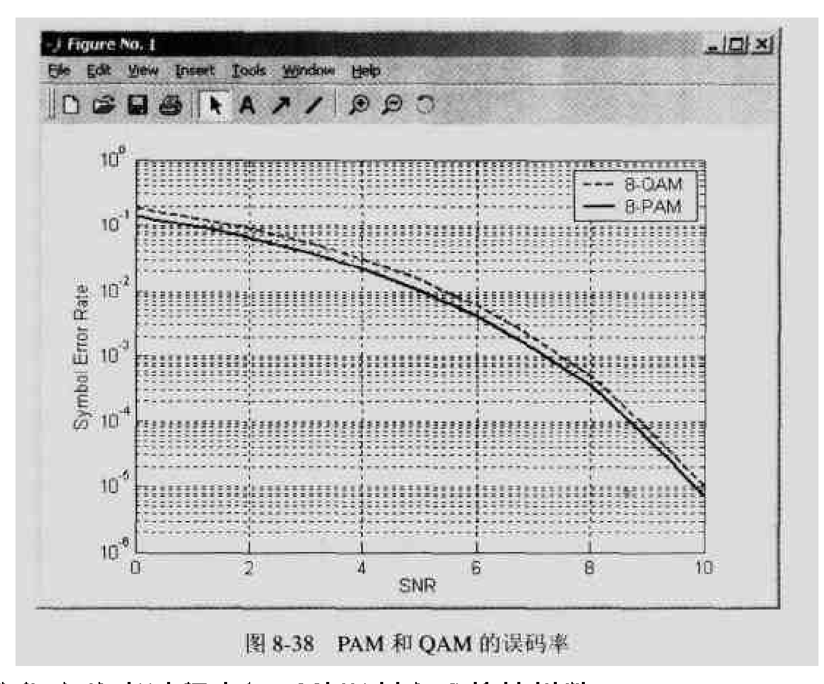



图 8-38　PAM 和 QAM 的误码率

另外，由于我们在仿真过程中把两种调制方式的抽样数（Samples per symbol 参数）设置为 1，因而仿真得到的误码率略高于理论计算数值。当增大 Samples per symbol 的数值时，PAM 和 QAM 的抗噪声性能随之增强，仿真得到的误码率也将降低，并且逐渐趋向于理论计算值。

## 8.5　数字频率调制

数字信号的频率调制通常称为“频移键控”（FSK，Frequency Shift Keying），由此产生的调制信号的频率随着输入信号的变化而相应地发生变化。M 相频移键控在调制过程中使用 $M$ 种不同的频率，频率的选择则取决于输入的数字信号的大小。在 MATLAB 中，频移键控的输入信号既可以是介于 0 和 $M$-1 之间的整数，也可以是一个二进制向量，这个二进制向量被看作是自然二进制序列或 Gray 码序列。频移键控产生的调制信号分成基带信号和频带信号两种类型，其中基带信号不包含高频载波，频带信号则是对基带信号进行频谱搬移的结果。本节将分别介绍基带频移键控信号和频带频移键控信号的调制和解调过程。

## 8.5.1 基带 M 相频移键控调制

M 相基带频移键控调制器（M-FSK Modulator Baseband）对输入信号实施 M 相频移键控调制，产生基带调制信号。在 M 相基带频移键控调制器中，参数 $M$ 表示调制过程中使用的频率的个数，这些频率之间的间隔是恒定的。M 相基带频移键控调制器模块及其参数设置对话框如图 8-39 所示。

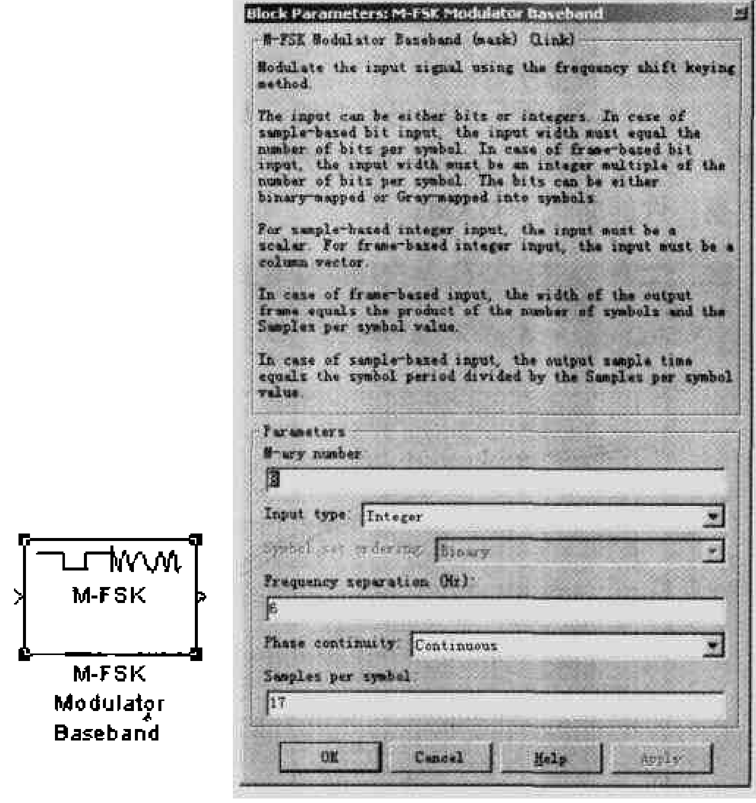



图 8-39 M 相基带频移键控调制器模块及其参数设置对话框

M 相基带频移键控调制器产生的基带调制信号的相位既可以是离散的，也可以是连续的，这取决于参数 Phase continuity 的设置。M 相基带频移键控调制器主要有以下几个参数。

■ M-ary number（M 相数）

M 相基带频移键控调制器在调制过程中使用的频率的个数。

■ Input type（输入信号类型）

M 相基带频移键控调制器输入信号的类型：当本参数设置为 Integer 时，每个输入信号是介于 0 和 $M$-1 之间的整数；当本参数设置为 Bit 时，输入信号是一个长度为 $k$ 的二进制向量，且 $k$ 满足条件 $M=2^k$。

■ Symbol set ordering（输入符号集编码方式）

当输入信号类型 Input type 设置为 Bit 时，本参数用于确定输入符号的编码方式：当本参数设置为 Binary 时，每个输入的长度为 $k$ 的二进制向量都当作是一个自然二进制序列编码；当本参数设置为 Gray 时，M 相基带频移键控调制器把长度为 $k$ 的二进制向量按照 Gray 编码序列进行处理。

■ Frequency separation (Hz)（频率间隔）

M 相基带频移键控信号相邻频率之间的间隔（单位：Hz）。

■　Phase continuity（相位连续性）

本参数用于确定 M 相基带频移键控信号的相位连续性：当本参数设置为 Continuous，输出信号的相位是连续的；当本参数设置为 Discontinuous 时，输出信号的相位则不一定连续。

■　Samples per symbol（输入符号抽样点的数目）

本参数表示每一个输入符号（整数或二进制向量）产生的 M 相基带频移键控输出信号的抽样点的个数。

M 相基带频移键控解调器（M-FSK Demodulator Baseband）对 M 相频移键控信号进行解调，得到原始的整数信号序列或二进制信号序列。M 相基带频移键控解调器模块及其参数设置对话框如图 8-40 所示。

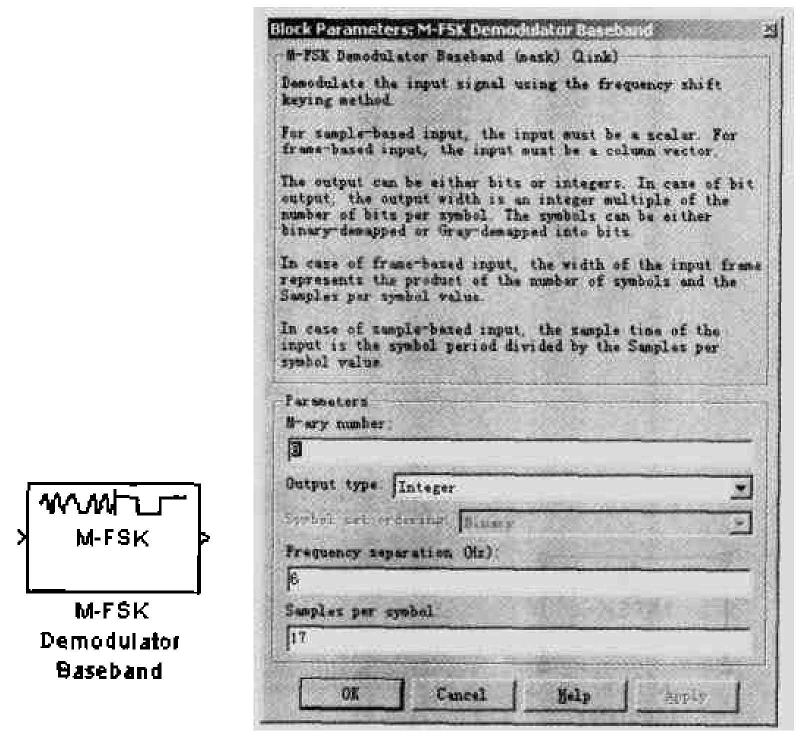



图 8-40　M 相基带频移键控解调器模块及其参数设置对话框

M 相基带频移键控解调器模块的参数 M 相基带频移键控调制器模块相同。

## 8.5.2　频带 M 相频移键控调制

M 相频带频移键控调制器（M-FSK Modulator Passband）对输入信号实施 M 相频移键控调制，产生频带调制信号。M 相频带频移键控调制器模块及其参数设置对话框如图 8-41 所示。

M 相频带频移键控调制器模块中的参数 M-ary number、Input type、Symbol set ordering、Frequency separation、Phase continuity 与 M 相基带频移键控调制器的同名参数相同。除此之外，M 相频带频移键控调制器还有以下几个参数。

■　Symbol period (s)（输入符号间隔）

M 相频带频移键控调制器每个输入符号（整数或二进制向量）的间隔（单位：秒）。

■　Baseband samples per symbol（基带符号抽样数）

M 相频带频移键控调制器每个输入符号（整数或二进制向量）的抽样数。

■ Carrier frequency (Hz)（载波频率）

M 相频带频移键控信号的载波的频率（单位：Hz）。

■ Carrier initial phase (rad)（载波相位）

M 相频带频移键控信号的载波的初始相位（单位：弧度）。

■ Output sample time(s)（输出信号抽样间隔）

M 相频带频移键控调制器输出信号的抽样间隔（单位：秒）。

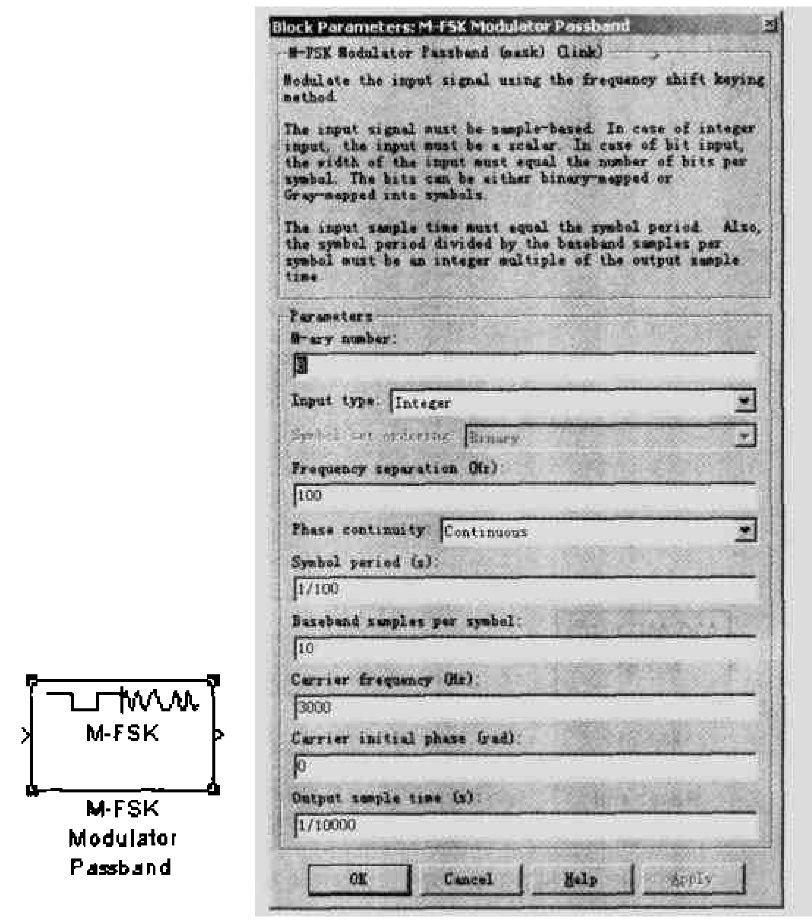



图 8-41　M 相频带频移键控调制器模块及其参数设置对话框

M 相频带频移键控解调器（M-FSK Demodulator Passband）对输入的 M 相频移键控调制信号实施解调，还原出原始的整数序列或二进制序列。M 相频带频移键控解调器模块及其参数设置对话框如图 8-42 所示。

M 相频带频移键控解调器模块的参数与 M 相频带频移键控调制器相同。

## 8.6 数字相位调制

数字信号的相位调制方式是根据输入的数字信号控制基带信号或频带信号的相位的一种调制方式，一般称为“相移键控”（PSK，Phase Shift Keying）。相移键控分为绝对相移键控和差分相移键控（DPSK，Differential Phase Shift Keying）两种，前者通过不同的相位来表示不同的数字信号，后者则用不同的相位差来表示不同的数字信号。在本节里，我们将依次介绍二进制相移键控 BPSK、四相相移键控 QPSK、偏移四相相移键控 OQPSK 以及 M 进制相

移键控 M-PSK 模块。

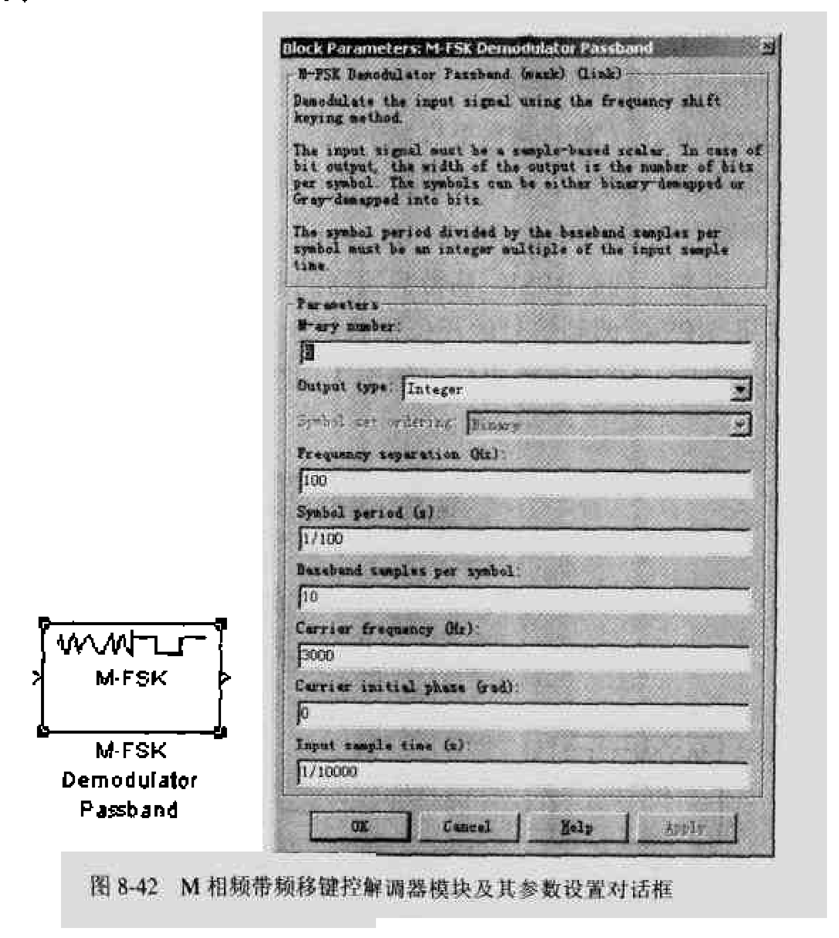



图 8-42　M 相频带频移键控解调器模块及其参数设置对话框

## 8.6.1　BPSK 调制

二进制相移键控（BPSK，Binary Phase Shift Keying）是用相位$\theta$和$\pi+\theta$分别表示二进制的 0 和 1 的一种数字调制方式。BPSK 基带调制器（BPSK Modulator Baseband）对输入的二进制信号（0 或 1）实施二进制相移键控调制，输出基带的调制信号。当输入信号是 0 时，输出信号等于$e^{j\theta}$；当输入信号等于 1 时，输出信号等于$-e^{j\theta}$，其中参数$\theta$表示相位偏移。BPSK 基带调制器模块及其参数设置对话框如图 8-43 所示。

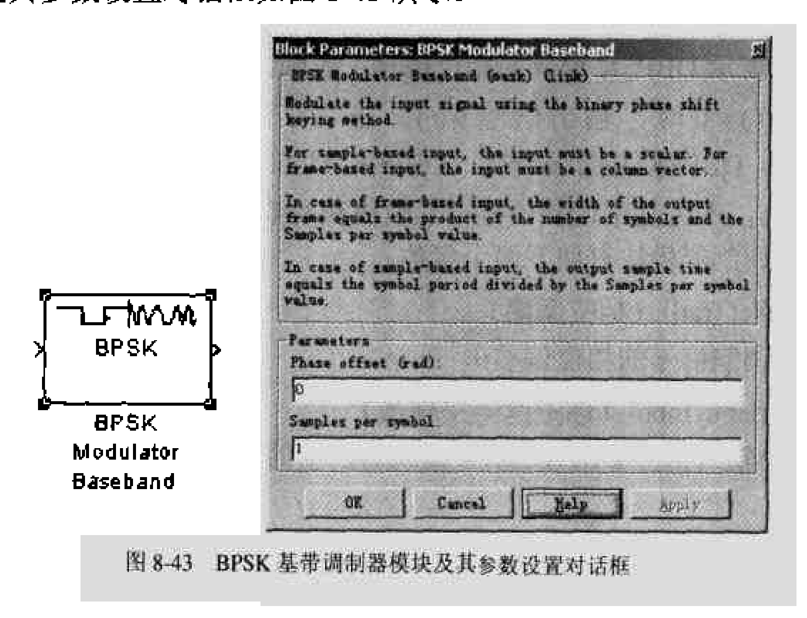



图 8-43　BPSK 基带调制器模块及其参数设置对话框

BPSK 基带调制器模块有两个参数。

■ Phase offset (rad)（相位偏移）

BPSK 基带调制信号的相位偏移$\theta$。

■ Samples per symbol（输出信号采样数）

对应于每一个输入信号 BPSK 基带调制器产生的输出信号的抽样点的个数。

BPSK 基带解调器（BPSK Demodulator Baseband）对二进制频移键控基带调制信号进行解调，得到原始的二进制序列。BPSK 基带解调器的输入信号可以是一个标量，也可以是帧格式的列向量。如果 BPSK 调制信号的相位偏移等于$\theta$，则当输入信号是$e^{j\theta}$时，输出信号等于 0；当输入信号等于$-e^{j\theta}$时，输出信号等于 1。图 8-44 所示是 BPSK 基带解调器模块及其参数设置对话框。

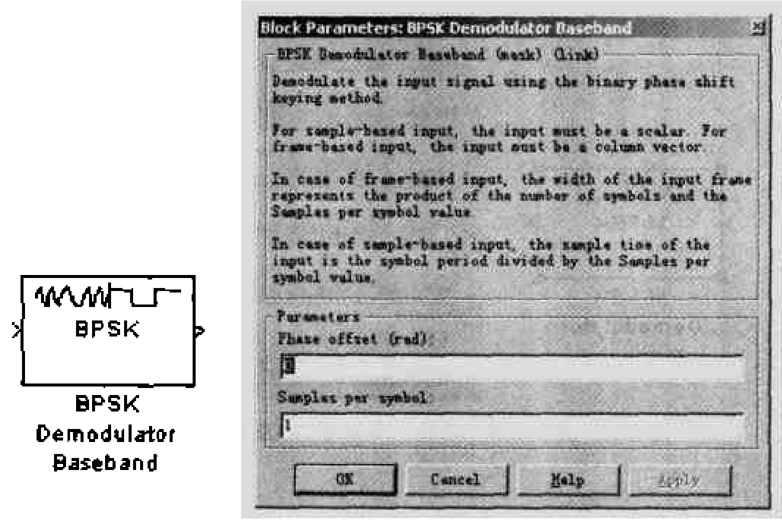



图 8-44　BPSK 基带解调器模块及其参数设置对话框

BPSK 基带解调器模块的参数与 BPSK 基带调制器模块相同。

## 8.6.2 DBPSK 调制

在差分二进制相位调制（DBPSK，Differential Binary Phase Shift Keying）中，假设输入信号为$x(t)$，输出信号为$y(t)$。对于第一个输入信号$x(1)$，如果$x(1)=0$，则输出信号$y(1)=e^{j\theta}$，其中$\theta$表示 DBPSK 调制的相位偏移；否则，$y(1)=-e^{j\theta}$。对于其后的每一个输入信号$x(t)$，当$x(t)=0$时，输出信号$y(t)=y(t-1)e^{j\theta}$；反之，当$x(t)=1$时，输出信号$y(t)=-y(t-1)e^{j\theta}$。DBPSK 基带调制器（DBPSK Modulator Baseband）对输入的二进制信号实施差分相位调制，产生基带调制信号，其模块和参数设置对话框如图 8-45 所示。

DBPSK 基带调制器模块有如下两个参数。

■ Phase offset (rad)（相位偏移）

DBPSK 基带调制信号的相位偏移$\theta$。

■ Samples per symbol（输出信号采样数）

对应于每一个输入信号 DBPSK 基带调制器产生的输出信号的抽样点的个数。

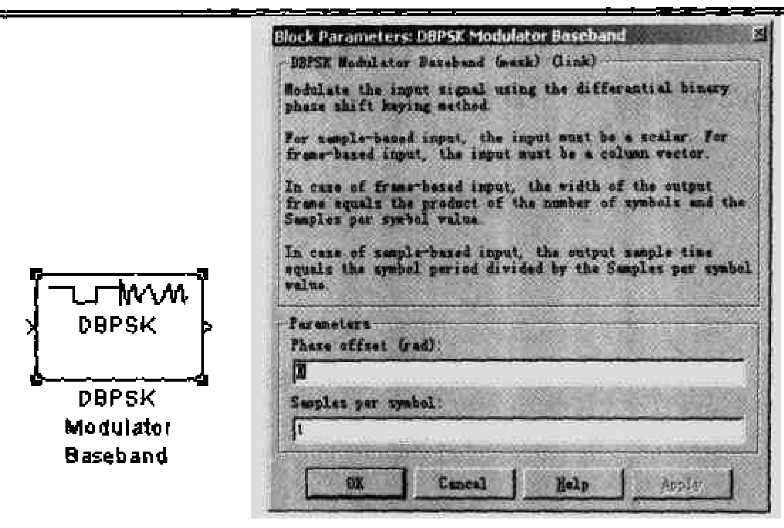



图 8-45　DBPSK 基带调制器模块及其参数设置对话框

DBPSK 基带解调器（DBPSK Demodulator Baseband）对差分二进制频移键控基带调制信号进行解调，得到原始的二进制序列。DBPSK 基带解调器的输入信号是一个复信号，它可以是一个标量，也可以是帧格式的列向量。图 8-46 所示是 DBPSK 基带解调器模块及其参数设置对话框。

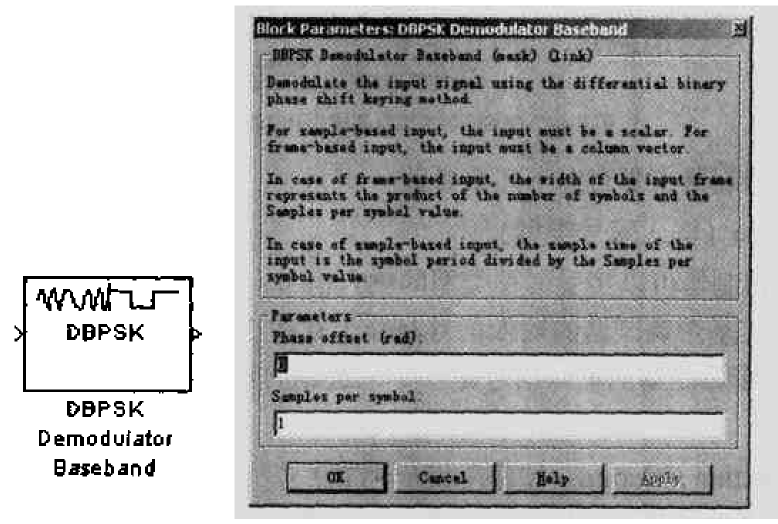



图 8-46　DBPSK 基带解调器模块及其参数设置对话框

假设 DBPSK 基带解调器的输入信号为 $y(t)$，输出信号为 $z(t)$，相位偏移为$\theta$。对于第一个输入信号 $y(1)$，如果 $y(1)=e^{j\theta}$，则输出信号 $z(1)=0$；否则，如果 $y(1)=-e^{j\theta}$，则 $z(1)=1$。对于其后的每一个输入信号 $y(t)$，当 $y(t)$ 和 $y(t-1)$ 的相位差等于$\theta$时，$z(t)=0$；反之，当 $y(t)$ 和 $y(t-1)$ 的相位相差 $\pi+\theta$ 时，输出信号 $z(t)=1$。DBPSK 基带解调器模块的参数与 DBPSK 基带调制器相同。

## 8.6.3　QPSK 调制

四相相移键控（QPSK，Quadrature Phase Shift Keying）是一种多进制数字相位调制方式它把输入信号 0、1、2、3 分别映射为 4 个不同的相位，这些相位之间的间隔等于$\pi/2$。假设 QPSK 调制的输入信号为 $x$，其中 $x$ 的取值范围是 0、1、2 或 3，则输出信号

$y=\exp(j\theta+jx\pi/2)$，其中 θ 是相位偏移。QPSK 基带调制器（QPSK Modulator Baseband）对输入信号实施 QPSK 调制，产生复数形式的基带调制信号，其模块框图和参数设置对话框如图 8-47 所示。

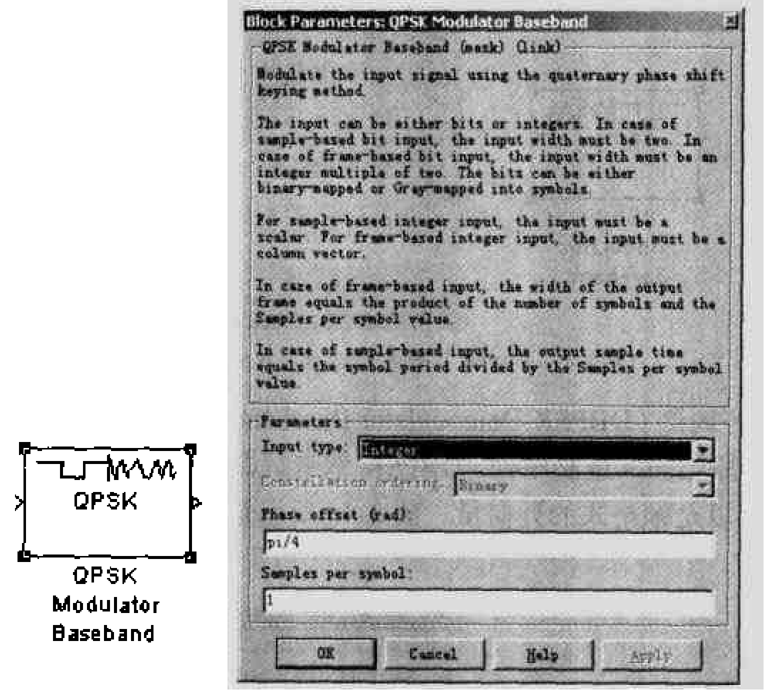



图 8-47　QPSK 基带调制器模块及其参数设置对话框

QPSK 基带调制器主要有以下几个参数。

■　Input type（输入信号类型）

QPSK 基带调制器可以接受两种类型的输入信号：Integer 或 Bit。当输入信号类型设置为 Integer 时，QPSK 基带调制器的输入信号是介于 0 和 3 之间的整数；当输入信号类型设置为 Bit 时，QPSK 基带调制器的输入信号是一个二进制向量，其中每两个二进制位表示一个四进制整数。

■　Constellation ordering（星座图编码方式）

当输入信号类型设置为 Bit 时，QPSK 基带调制器输入信号中的每两个二进制位表示一个四进制整数，这两个二进制位既可以是自然二进制编码序列，也可以是 Gray 码序列，它们产生不同的星座图。当 Constellation ordering 设置为 Binary 时，输入信号是自然二进制编码序列；当 Constellation ordering 设置为 Gray 时，输入信号是 Gray 码序列。图 8-48 所示是当相位偏移 $\theta=\pi/4$ 时自然二进制编码和 Gray 码产生的不同的星座图。

■　Phase offset (rad)（相位偏移）

QPSK 基带调制的相位偏移 $\theta$（单位：弧度）。

■　Samples per symbol（输出信号采样数）

对应于每一个输入信号 QPSK 基带调制器产生的输出信号的抽样点的个数。

QPSK 基带解调器（QPSK Demodulator Baseband）对输入的 QPSK 基带调制信号进行解调，这个输入信号是标量形式或帧格式的复信号。假设 QPSK 解调器的输入信号为 $y$，相位偏移等于$\theta$，则当 $y=\exp(j\theta+jm\pi/2)$ 时，输出信号 $z=m$。QPSK 基带解调器的输出信号既

可以是整数，也可以是一个长度为 2 的二进制向量，其模块框图和参数设置对话框如图 8-49 所示。

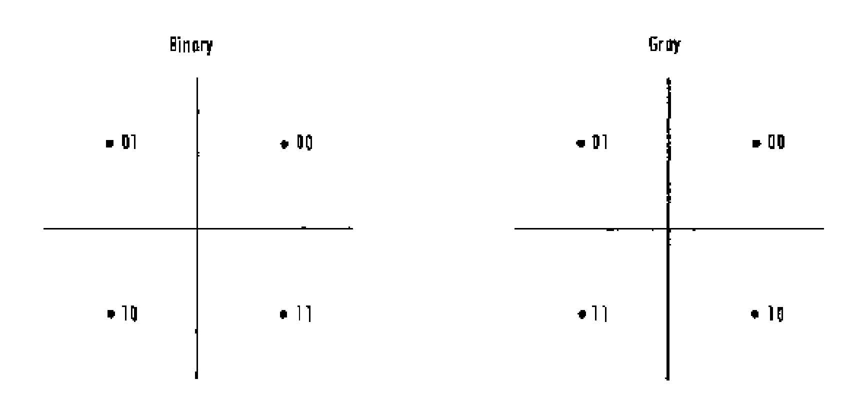



图 8-48　自然二进制编码和 Gray 码的星座图

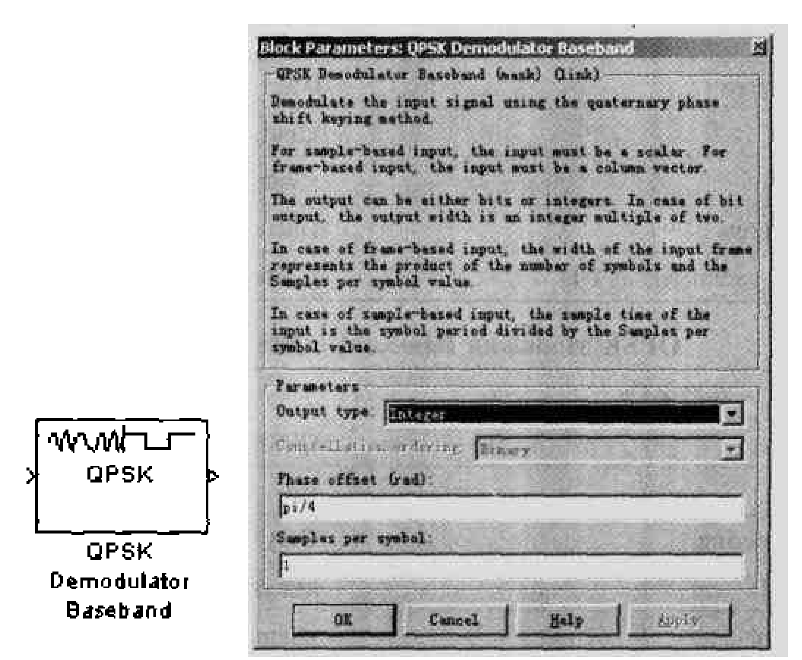



图 8-49　QPSK 基带解调器模块及其参数设置对话框

QPSK 基带解调器模块的参数与 QPSK 基带调制器相同。

### 8.6.4　实例 8.2——QPSK 在 IS-95 前向信道中的应用

在 IS-95 系统中，前向信道（从基站到移动台方向的信道）采用 QPSK 调制方式对两个支路的二进制信号进行调制。本节将设计一个 QPSK 仿真模型，以衡量 QPSK 在高斯白噪声信道中的性能。

图 8-50 所示是 QPSK 调制的仿真模型。在这个仿真模型中，Bernoulli Binary Generator（贝努利二进制序列产生器）产生一个二进制向量，向量的长度等于 2，分别代表 QPSK Modulator Baseband（QPSK 调制器）的两个输入信号。表 8-8 和表 8-9 所示列出了 Bernouli Binary Generator 和 QPSK Modulator Baseband 的参数设置。

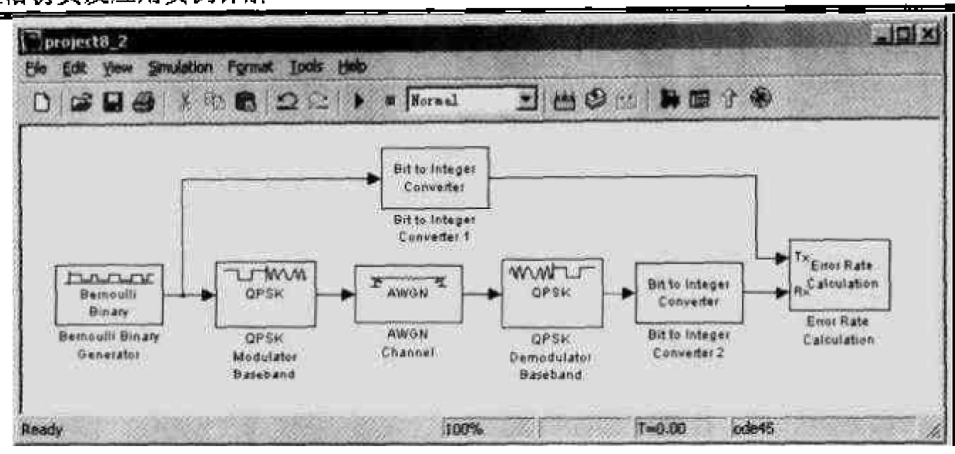



图 8-50 QPSK 调制的仿真模型

表 8-8 Bernoulli Binary Generator（贝努利二进制序列产生器）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | xInitialSeed |
| Sample time | xSampleTime |
| Frame-based outputs | Unchecked |
| Interlpret vector parameters as 1-D | Unchecked |

表 8-9 QPSK Modulator Baseband（QPSK 基带调制器模块）的参数设置

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | QPSK Modulator Baseband |
| Input type | Bit |
| Constellation ordering | Gray |
| Phase offset (rad) | xPhaseOffset |
| Samples per symbol | xSamplesPerSymbol |

IS-95 对前向信道进行 QPSK 调制的过程可以看作是一种相位映射过程，即把两个支路（I 支路和 Q 支路）的信号映射为调制信号相位，这种映射关系如表 8-10 所示。

表 8-10 IS-95 前向信道 QPSK 调制的相位映射

| I 支路数据 | Q 支路数据 | 相位 |
| --- | --- | --- |
| 0 | 0 | $\pi/4$ |
| 1 | 0 | $3\pi/4$ |
| 1 | 1 | $-3\pi/4$ |
| 0 | 1 | $-\pi/4$ |

从表 8-10 中可以看到，如果把 Q 支路数据看作是 QPSK 调制器模块第一个输入端口的输入信号，而把 I 支路数据看作是第二个输入端口的输入信号，那么 IS-95 前向信道的这种映射关系实际上对应于 QPSK 调制器模块相位偏移等于 $\pi/4$ 时的 Gray 星座图。因此，我们把 QPSK 调制器模块的 Constellation ordering 参数设置为 Gray。

QPSK 调制器产生的调制信号首先经过一个 AWGN Channel（加性高斯白噪声信道），然

后进入 QPSK Demodulator Baseband（QPSK 解调器模块）进行解调。表 8-11 和表 8-12 列出了这两个模块的参数设置，其中调制信号经过信道之后的信噪比由变量 xSNR 确定，在后面的仿真过程中我们通过改变这个变量的数值得到相应的误码率，并且根据这两者的关系绘制 QPSK 调制的误码率曲线。

**表 8-11　AWGN Channel（加性高斯白噪声模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | AWGN Channel |
| Initial seed | 67 |
| Mode | Signal to noise ratio (SNR) |
| SNR (dB) | xSNR |
| Input signal power (watts) | 1 |

**表 8-12　QPSK Demodulator Baseband（QPSK 基带解调器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | QPSK Demodulator Baseband |
| Output type | Bit |
| Constellation ordering | Gray |
| Phase offset (rad) | xPhaseOffset |
| Samples per symbol | xSamplesPerSymbol |

由于贝努利二进制序列产生器的输出信号是长度为 2 的二进制向量，而 QPSK 基带解调器模块的输出信号则是一个二进制序列，因此在对它们进行比较之前，首先通过两个数值转换模块（Bit to Integer Converter 1 和 Bit to Integer Converter 2）把它们转换成四进制整数。这两个数值转换模块的参数设置如表 8-13 所示。

**表 8-13　Bit to Integer Converter（数值转换模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bit to Integer Converter |
| Number of bits per integer | 2 |

最后两个数值转换模块的输出信号进入一个误码率统计模块（Error Rate Calculation），以统计 QPSK 解调信号的误码率。表 8-14 列出了误码率统计模块的参数设置。需要注意的是，当 QPSK 调制器和解调器中的 Samples per symbol 参数大于 1 时，解调信号将落后调制前的信号一个周期，在这里我们用变量 xReceiveDelay 来表示这个时延。同时，我们把仿真模型的运行时间设置为 xSimulationTime，这个变量的数值将在脚本程序中定义。

**表 8-14　Error Rate Calculation（误码率统计模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Error Rate Calculation |
| Receive delay | xReceiveDelay |
| Computation delay | 0 |
| Computation mode | Entire frame |

续表

| 参数名称 | 参数值 |
| --- | --- |
| Output data | Workspace |
| Variable name | xErrorRate |
| Reset port | Unchecked |
| Stop simulation | Unchecked |

为了得到 QPSK 调制信号误码率与信号的信噪比之间的关系曲线，我们编写了一个 M 文件 project8_2main.m，其代码如下：

```
% 设置调制信号的抽样间隔
xSampleTime = 1/100000;
% 设置仿真时间的长度
xSimulationTime = 10;
% 设置随机数产生器的初始化种子
xInitialSeed = [61 71];
% 设置 QPSK 调制的初始相位
xPhaseOffset = pi/4;

% x 表示信噪比的取值范围
x = 0:10;
% y 表示 QPSK 调制的误符号率
y = x;

hold off;
for index = 1:4
    % 设置 QPSK 调制信号的抽样个数
    xSamplesPerSymbol = index;
    % 设置绘图颜色和误码率计算模块的接收时延
    % 当 xSamplesPerSymbol 不等于 1 时误码率计算模块有一个符号的接收时延
    switch index
        case 1
            xReceiveDelay = 0;
            color = 'r';
        case 2
            xReceiveDelay = 1;
            color = 'g';
        case 3
            xReceiveDelay = 1;
            color = 'b';
        case 4
```

```
            xReceiveDelay = 1;
            color = 'm';
    end
    for i = 1:length(x)
        % 信噪比依次取向量 x 的数值
        xSNR = x(i);
        % 执行 QPSK 仿真模型
        sim('project8_2');
        % 从 xErrorRate 中获得调制信号的误码率
        y(i) = xErrorRate(1);
    end
    % 绘制信噪比与误码率的关系曲线
    semilogy(x,y,color);
    hold on;
end
```

在 MATLAB 工作区中输入命令“project8_2main”，仿真程序开始运行，运行时间的长短取决于变量 xSampleTime 和 xSimulationTime 的数值。仿真结束之后我们得到图 8-51 所示的曲线图。

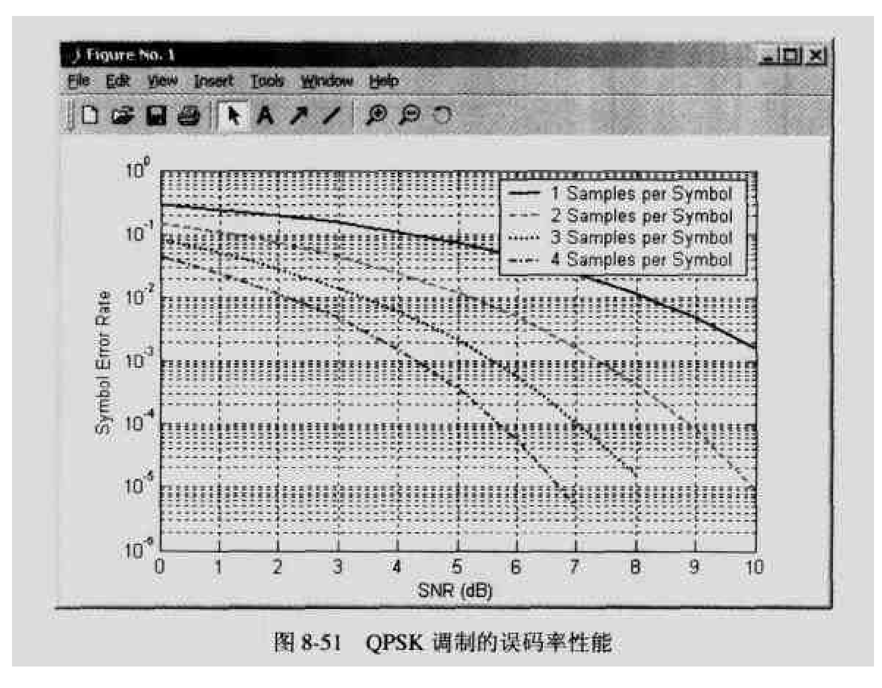



图 8-51　QPSK 调制的误码率性能

在这个仿真模型中，我们绘制了 4 条曲线，分别表示当 QPSK 调制器和解调器的 Samples per symbol 参数等于 1、2、3、4 时 QPSK 信号的误码率性能。从图中可以看到，Samples per symbol 参数的设置在很大程度上影响着 QPSK 信号的解调性能。例如，当信噪比等于 7 分贝时，如果 Samples per symbol 等于 1，QPSK 调制信号的误码率大约等于 2.5%；如果 Samples per symbol 设置为 4，这时候误码率大约只有 0.0005%。

### 8.6.5 DQPSK 调制

差分四相相位调制（DQPSK，Differential Quadrature Phase Shift Keying）是差分形式的 QPSK 调制。假设输入信号为$x(t)$，输出信号为$y(t)$，其中$x(t)\in\{0\quad 1\quad 2\quad 3\}$。对于第一个输入信号$x(1)=m$，输出信号$y(1)=\exp(j\theta+jm\pi/2)$，其中$\theta$表示 DQPSK 调制的相位偏移；对于其后的每一个输入信号$x(t)=m$，输出信号$y(t)=y(t-1)\exp(j\theta+jm\pi/2)$。DQPSK 基带调制器（DQPSK Modulator Baseband）对输入的四进制信号实施 DQPSK 调制，产生复数形式的基带调制信号，其模块和参数设置对话框如图 8-52 所示。

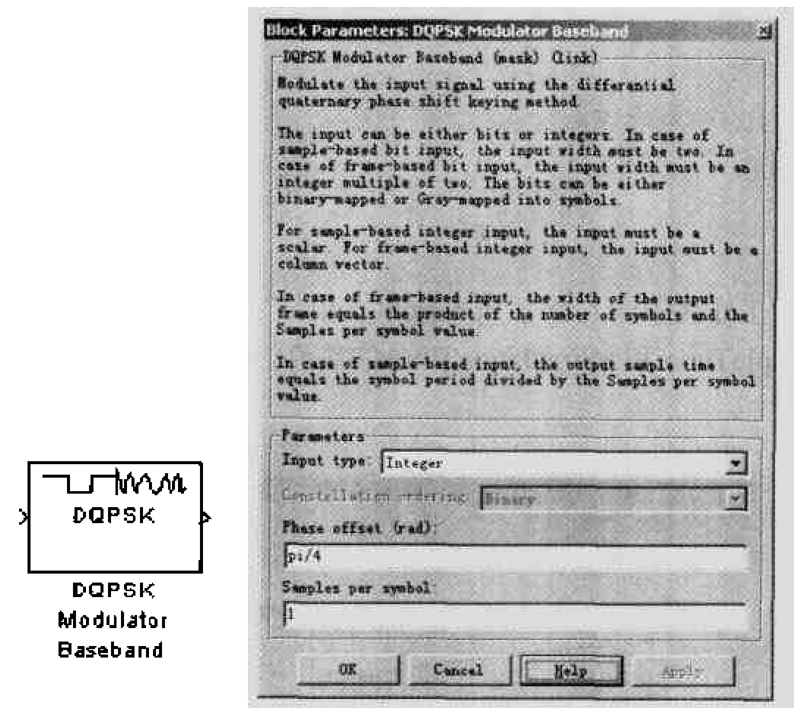



图 8-52　DQPSK 基带调制器模块及其参数设置对话框

对于 DQPSK 基带调制器，当它的相位偏移等于$\theta$等于$\pi/k$时，其中$k$是一个整数，则 DQPSK 调制信号的星座图中有$2k$个点。例如，当$\theta=\pi/4$时，DQPSK 的星座图中有 8 个点。图 8-53 所示是$\theta=\pi/4$时产生的基带调制信号的轨迹图，图中的连线表示星座图中的两个点之间存在着转换关系。

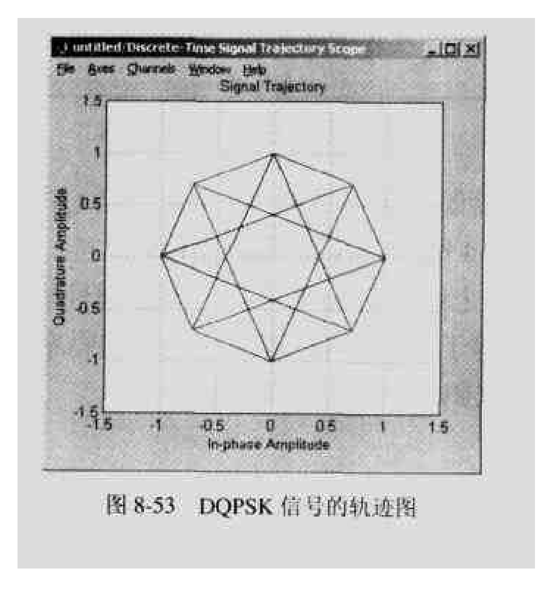



图 8-53　DQPSK 信号的轨迹图

DQPSK 基带调制器主要有以下几个参数。

■　Input type（输入信号类型）

本参数用于确定 DQPSK 基带调制器的输入信号类型。当 Input type 设置为 Integer 时，DQPSK 基带调制器的输入信号是介于 0 和 3 之间的整数；当 Input type 设置为 Bit 时，DQPSK 基带调制器的输入信号是一个二进制向量，其中每两个二进制位表示一个四进制整数。

■　Constellation ordering（星座图编码方式）

当输入信号类型设置为 Bit 时，本参数用于确定 DQPSK 基带调制器输入信号的编码方式：当 Constellation ordering 设置为 Binary 时，输入信号是自然二进制编码序列；当 Constellation ordering 设置为 Gray 时，输入信号是 Gray 码序列。这两种编码方式产生如图 8-48 所示的两种星座图。

■　P hase offset (rad)（相位偏移）

DQPSK 基带调制的相位偏移$\theta$（单位：弧度）。

■　Samples per symbol（输出信号采样数）

对应于每一个输入信号 DQPSK 基带调制器产生的输出信号的抽样点的个数。

DQPSK 基带解调器（DQPSK Demodulator Baseband）对 DQPSK 基带调制信号进行解调。假设 DQPSK 基带解调器的输入信号为$y(t)$，输出信号为$z(t)$，相位偏移为$\theta$。对于第一个输入信号$y(1)$，如果$y(1)=\exp(j\theta+jm\pi/2)$，则输出信号$z(1)=m$；对于其后的每一个输入信号$y(t)$，当$y(t)$和$y(t-1)$的相位差等于$\theta+m\pi/2$时，$z(t)=m$。DQPSK 基带解调器的输出信号既可以是整数，也可以是一个长度为 2 的二进制向量，其模块框图和参数设置对话框如图 8-54 所示。

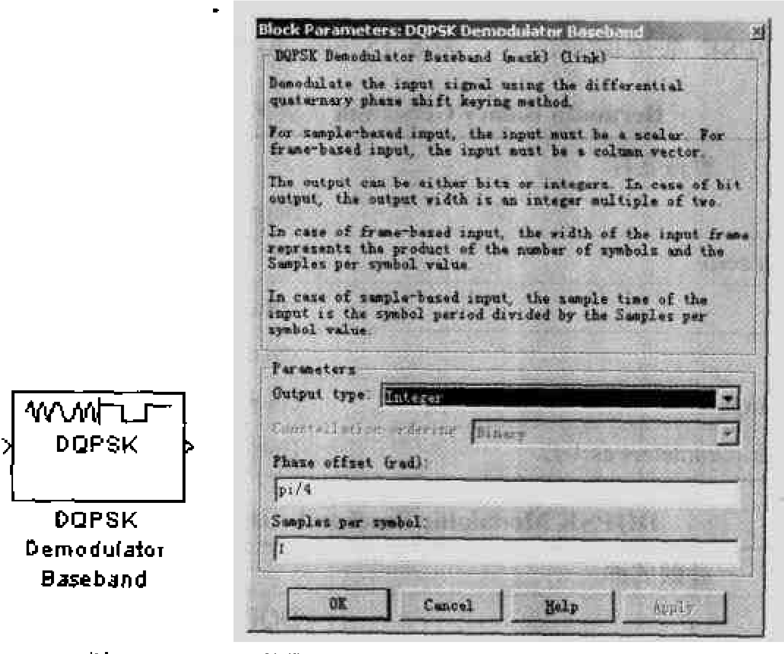



图 8-54　DQPSK 基带解调器模块及其参数设置对话框

DQPSK 基带解调器模块的参数与 DQPSK 基带调制器模块相同。

## 8.6.6　实例 8.3——DQPSK 在 USDC 中的应用

美国数字蜂窝系统（USDC，United States Digital Cellular system）是继 AMPS 之后的又一种蜂窝通信系统，它采用数字调制方式，因此又被称为数字 AMPS 系统（D-AMPS）。由于采用了数字调制方式，USDC 系统的容量是 AMPS 的六倍，在很大程度上满足了移动用户数

量快速增长的要求。

USDC 系统的话音信道采用$\pi/4$ DQPSK 调制方式，信道传输速率可以达到 48.6kbit/s。除此之外，$\pi/4$ DQPSK 还应用于 PACS（Personal Access Communication System）和 PHS（Personal Handyphone System）中。本节我们设计一个 DQPSK 调制和解调系统的仿真模型，以考察 DQPSK 调制信号的抗噪声性能，并且与 QPSK 调制信号的误码率进行比较。图 8-55 所示是这个仿真模型的系统结构。

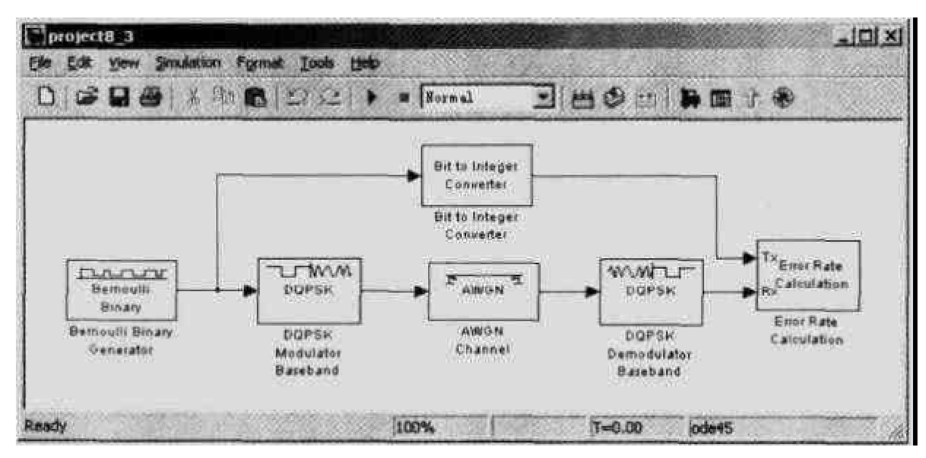



图 8-55 DQPSK 调制的仿真模型

在 DQPSK 调制的仿真模型中，Bernoulli Binary Generator（贝努利二进制序列产生器）产生一个二进制向量，这个二进制向量的长度等于 2，分别代表 DQPSK 调制器两个支路的输入信号。我们采用 DQPSK Modulator Baseband（DQPSK 基带调制器模块）对这个信号进行调制，产生 DQPSK 基带调制信号。表 8-15 和表 8-16 列出了这两个模块的参数设置情况。

表 8-15 **Bernoulli Binary Generator（贝努利二进制序列产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | xInitialSeed |
| Sample time | xSampleTime |
| Frame-based outputs | Unchecked |
| Interlpret vector parameters as 1-D | Unchecked |

表 8-16 **DQPSK Modulator Baseband（DQPSK 基带调制器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | DQPSK Modulator Baseband |
| Input type | Bit |
| Constellation ordering | Binary |
| Phase offset (rad) | xPhaseOffset |
| Samples per symbol | xSamplesPerSymbol |

与实例 8.2 类似地，基带调制信号首先经过一个 AWGN Channel（加性高斯白噪声模块），然后由 QPSK Demodulator Baseband（QPSK 基带解调器模块）对其实施解调。这个 QPSK 基带解调器的参数设置与 QPSK 基带调制器保持一致，它们具有相同的相位偏移和抽样个数，

并且输出整数形式的解调信号。表 8-17 和表 8-18 是这两个模块的参数设置。

**表 8-17　　AWGN Channel（加性高斯白噪声模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | AWGN Channel |
| Initial seed | 67 |
| Mode | Signal to noise ratio (SNR) |
| SNR (dB) | xSNR |
| Input signal power (watts) | 1 |

**表 8-18　　DQPSK Demodulator Baseband（DQPSK 基带解调器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | DQPSK Demodulator Baseband |
| Output type | Integer |
| Phase offset (rad) | xPhaseOffset |
| Samples per symbol | xSamplesPerSymbol |

贝努利二进制序列产生器的输出信号在通过 Bit to Integer Converter（数值转换模块）转换成整数之后，与 QPSK 解调信号一起进入 Error Rate Calculation（误码率统计模块），这两个模块的参数设置如表 8-19 和表 8-20 所示。误码率统计模块信号接收端的时延等于 xReceiveDelay，同时它把误码率的统计结果保存在工作区变量 xErrorRate 中。

**表 8-19　　Bit to Integer Converter（数值转换模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bit to Integer Converter |
| Number of bits per integer | 2 |

**表 8-20　　Error Rate Calculation（误码率统计模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Error Rate Calculation |
| Receive delay | xReceiveDelay |
| Computation delay | 0 |
| Computation mode | Entire frame |
| Output data | Workspace |
| Variable name | xErrorRate |
| Reset port | Unchecked |
| Stop simulation | Unchecked |

M 文件 project8_3main.m 对仿真模型中的各个变量进行赋值，然后依次改变信号的信噪比，循环执行仿真程序，并且在相同的条件下对实例 8.2 的模型进行仿真，最后根据仿真的结果绘制曲线。M 文件 project8_3main.m 的代码如下：

```
% 设置调制信号的抽样间隔
xSampleTime = 1/100000;
```

```
% 设置仿真时间的长度
xSimulationTime = 10;
% 设置随机数产生器的初始化种子
xInitialSeed = [61 71];
% 设置 DQPSK 调制的初始相位
xPhaseOffset = pi/4;
% 设置 DQPSK 调制信号的抽样个数
xSamplesPerSymbol = 1;
% 设置误码率统计模块的接收时延
xReceiveDelay = 0;

% x 表示信噪比的取值范围
x = 0:10;
% y1 表示 DQPSK 调制的误符号率
y1 = x;
% y2 表示 QPSK 调制的误符号率
y2 = x;

for i = 1:length(x)
    % 信噪比依次取向量 x 的数值
    xSNR = x(i);
    % 执行 DQPSK 仿真模型
    sim('project8_3');
    % 从 xErrorRate 中获得调制信号的误码率
    y1(i) = xErrorRate(1);
    % 执行 QPSK 仿真模型
    sim('project8_2');
    % 从 xErrorRate 中获得调制信号的误码率
    y2(i) = xErrorRate(1);
end

% 绘制信噪比与误码率的关系曲线
semilogy(x,y1,'r',x,y2,'b');
```

要运行仿真程序，只需在 MATLAB 工作区中输入命令行“project8_3”。在信噪比较高的条件下，如果仿真的时间不够长，这时候仿真得到的误比特率通常等于零。为此，我们把贝努利二进制序列产生器的抽样间隔 xSampleTime 设置为 1/100000，同时把仿真时间 xSimulationTime 设置为 10 秒，从而在一个仿真循环中产生 $10^6$ 个调制信号，以此提高仿真数据的精度。由此带来的另外一个问题是仿真需要较长的执行时间，这个时间还取决于计算机的运算速度。仿真结束之后我们得到如图 8-56 所示的误码率曲线。

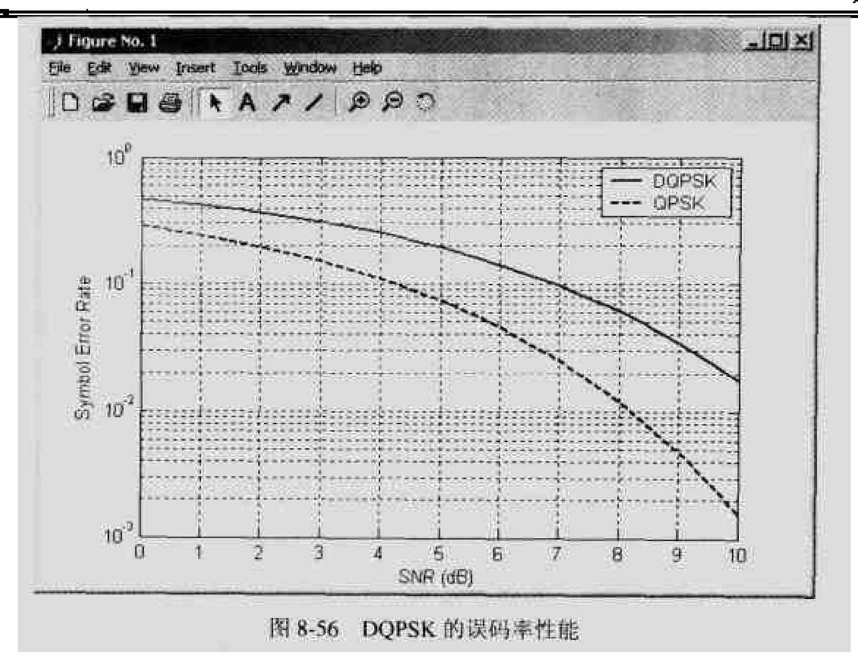



图 8-56　DQPSK 的误码率性能

在图 8-56 所示中，虚线表示实例 8.2 中 QPSK 调制信号的误码率性能，实线则是本节介绍的 DQPSK 调制信号的误码率。从图中可以看到，在相同的条件下（相同的数据源、相同的信噪比以及相同的调制信号抽样数），QPSK 调制信号的性能优于 DQPSK。

## 8.6.7　基带 OQPSK 调制

在 QPSK 调制中，当输入的二进制信号由 00 变成 11 或由 01 变成 10 时，QPSK 调制信号的相位翻转±180°，这种翻转在很大程度上削弱了 QPSK 调制的抗噪声性能。作为 QPSK 调制的一种改进方式，偏移四相相移键控（OQPSK，Offset Quadrature Phase Shift Keying）产生的调制信号的相位至多只能翻转±90°。在 OQPSK 调制信号 $y(t)$ 中，Q 支路信号 $y_Q(t)$ 落后 I 支路信号 $y_I(t)$ 半个符号周期。这样，OQPSK 调制信号中的 I 支路信号和 Q 支路信号不可能同时改变数值，从而消除了±180° 相位翻转。图 8-57 所示是当相位偏移 $\theta=0$ 时 OQPSK 输入信号和调制信号的波形，两根线条分别表示 I 支路信号和 Q 支路信号。从图中可以看出，OQPSK 调制信号可以看作是 QPSK 调制信号的 Q 支路延迟半个符号周期输出的结果。

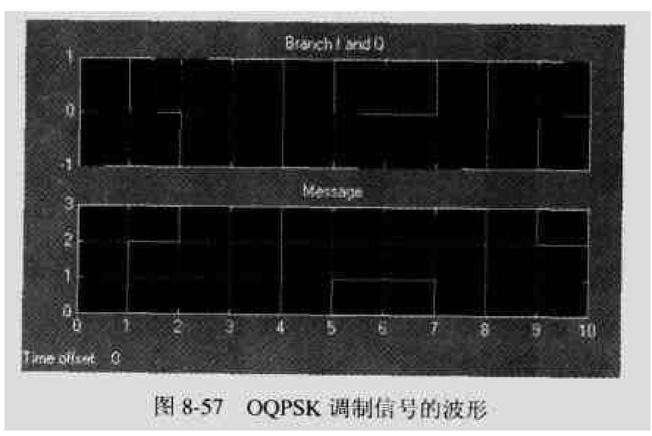



图 8-57　OQPSK 调制信号的波形

需要注意的是，在 MATLAB 中，对于相同的相位偏移 $\theta$，OQPSK 调制信号的星座图与 QPSK 调制信号不同，图 8-58 所示是当 $\theta=\pi/4$ 时 OQPSK 调制的输入信号与输出相位之间的对应关系。

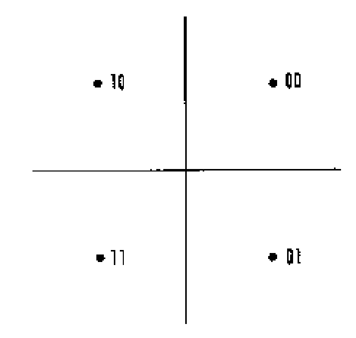



图 8-58 OQPSK 基带调制器输入信号与输出相位的对应关系

由于 OQPSK 调制信号 $y(t)$ 的 Q 支路信号 $y_Q(t)$ 落后 I 支路信号 $y_I(t)$ 半个符号周期，因此对于同一个输入信号 $x(t)$，输出信号 $y_I(t)$ 和 $y_Q(t)$ 可以分成两个部分，即 $y_I^{(1)}(t)$ 和 $y_I^{(2)}(t)$，以及 $y_Q^{(1)}(t)$ 和 $y_Q^{(2)}(t)$，它们分别对应于一个输入信号的前半个符号周期和后半个符号周期的输出信号，它们满足如下关系式：

$$y_I^{(1)}(t)=y_I^{(2)}(t)=\begin{cases}\cos\theta & x(t)=00\\ \sin\theta & x(t)=01\\ -\sin\theta & x(t)=10\\ -\cos\theta & x(t)=11\end{cases} \tag{8.2}$$

$$y_Q^{(2)}(t)=\begin{cases}\sin\theta & x(t)=00\\ -\cos\theta & x(t)=01\\ \cos\theta & x(t)=10\\ -\sin\theta & x(t)=11\end{cases} \tag{8.3}$$

$$y_Q^{(1)}(t)=y_Q^{(2)}(t-1) \tag{8.4}$$

从上面的 3 个公式中可以看到，OQPSK 调制信号即 $y_I^{(1)}(t)$、$y_I^{(2)}(t)$、$y_Q^{(1)}(t)$ 以及 $y_Q^{(2)}(t)$ 的取值与相位偏移 $\theta$ 有关，并且它们只能取 $\pm\sin\theta$ 和 $\pm\cos\theta$ 四个数值中的一个。在任意时刻，OQPSK 调制信号等于 $y_I^{(1)}(t)+i\times y_Q^{(1)}(t)$ 或 $y_I^{(2)}(t)+i\times y_Q^{(2)}(t)$，因此，OQPSK 调制的星座图中至多只有 16 个点。

OQPSK 基带调制器（OQPSK Modulator Baseband）对输入信号实施 OQPSK 调制，产生复数形式的基带信号。OQPSK 基带调制器模块及其参数设置对话框如图 8-59 所示。

OQPSK 基带调制器有以下 3 个参数。

■ Input type（输入信号类型）

当输入信号类型设置为 Integer 时，OQPSK 基带调制器的输入信号是介于 0 和 3 之间的整数；当输入信号类型设置为 Bit 时，OQPSK 基带调制器的输入信号是一个二进制向量，其中每两个二进制位表示一个四进制整数。

■ Phase offset (rad)（相位偏移）

OQPSK 基带调制的相位偏移 $\varphi$（单位：弧度），它是相对于 $\pi/4$ 相位的偏移值，即 $\phi=\theta-\pi/4$。

■ Samples per symbol（输出信号采样数）

对应于每一个输入信号 OQPSK 基带调制器产生的输出信号的抽样点的个数。

图 8-60 所示是当相位偏移$\theta$ 分别等于$\pi/8$和$\pi/4$时 OQPSK 调制信号的发散图。从图中可以看到，当$\theta=\pi/8$时，星座图中共有 16 个点；而当$\theta=\pi/4$时，星座图中只有 4 个点。造成这种现象的原因在于，当$\theta=\pi/4$时，$\sin\theta=\cos\theta=\sqrt{2}/2$，使得 OQPSK 的 I 支路调制信号和 Q 支路调制信号的取值只能是$\sqrt{2}/2$和$-\sqrt{2}/2$两个值中的一个。

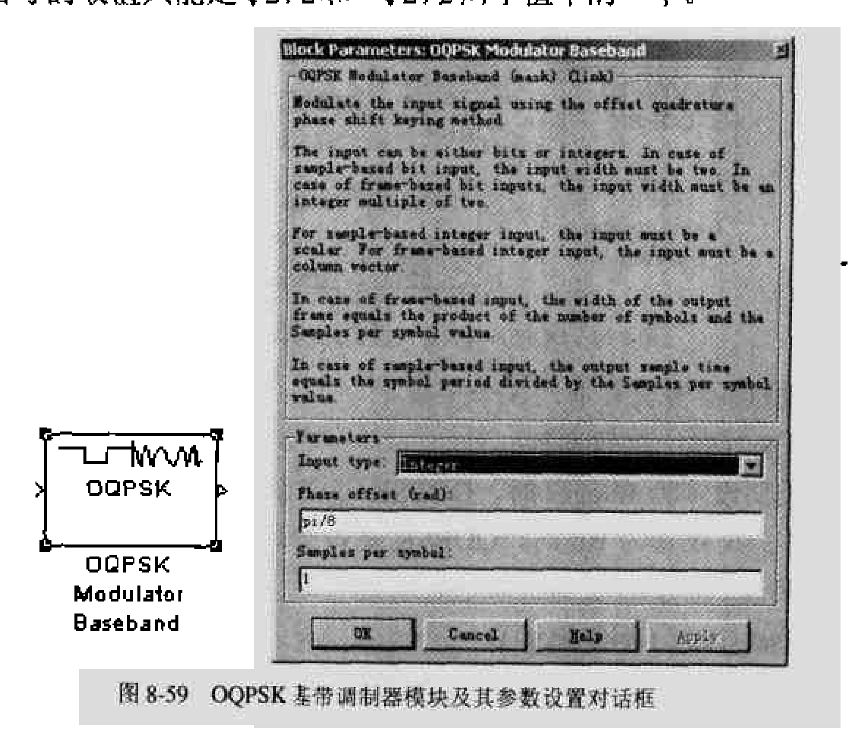



图 8-59 OQPSK 基带调制器模块及其参数设置对话框

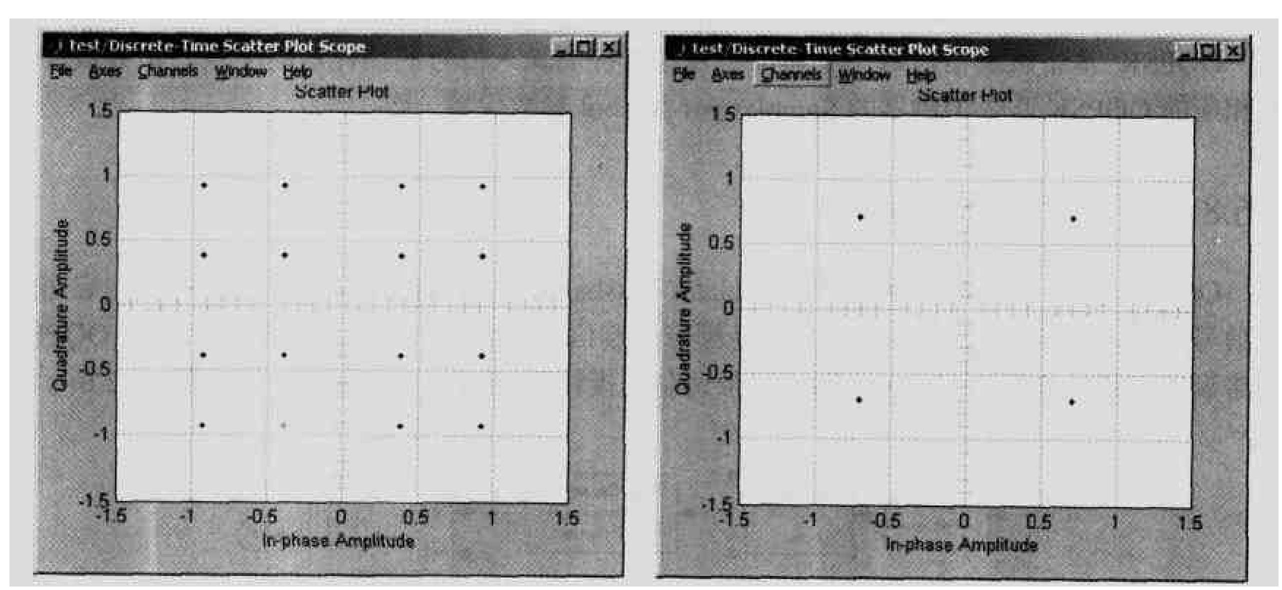



图 8-60 OQPSK 调制信号的星座图

OQPSK 基带解调器对 OQPSK 基带调制信号进行解调，还原得到调制前的整数序列或二进制序列。OQPSK 基带解调器模块及其参数设置对话框如图 8-61 所示。

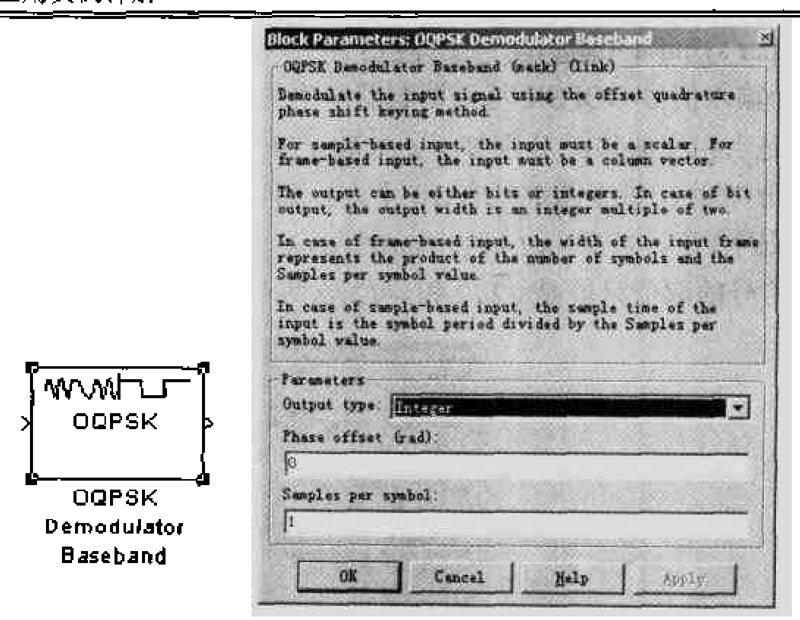



图 8-61　OQPSK 基带解调器模块及其参数设置对话框

OQPSK 基带解调器模块有下列 3 个参数。

■　Output type（输出信号类型）

当输出信号类型设置为 Integer 时，OQPSK 基带解调器的输出信号是介于 0 和 3 之间的整数；当输出信号类型设置为 Bit 时，OQPSK 基带解调器的输出信号是一个二进制向量，其中每两个二进制位表示一个四进制整数。

■　Phase offset (rad)（相位偏移）

OQPSK 基带调制信号的相位偏移 $\varphi$（单位：弧度），它是相对于 $\pi/4$ 相位的偏移值，即 $\varphi=\theta-\pi/4$。

■　Samples per symbol（输入信号采样数）

OQPSK 基带解调器与每一个输出符号相对应的输入信号的抽样点的个数。本参数应该与相应的 OQPSK 基带调制器的 Samples per symbol 参数保持一致。

## 8.6.8　频带 OQPSK 调制

OQPSK 频带调制器（OQPSK Modulator Passband）与 OQPSK 基带调制器相似，它对输入信号实施 OQPSK 调制，但是产生的是频带调制信号，并且这个输出信号是实信号。OQPSK 频带调制器模块实际上是由 OQPSK 基带调制器模块构成的，它的内部结构如图 8-62 所示。

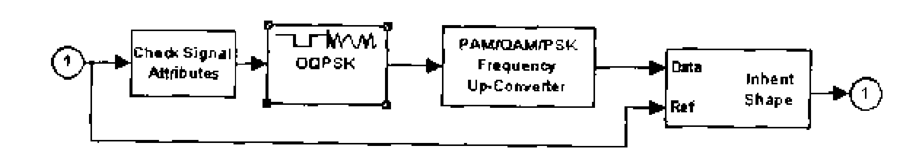



图 8-62　OQPSK 频带调制器模块的内部结构

从图 8-62 中可以看到，在 OQPSK 频带调制过程中，输入信号首先被 OQPSK 基带调制器转换成基带调制信号，然后把这个基带信号调制到载频上去，从而得到频带调制信号。值得注意的是，其中的 OQPSK 基带调制器的相位偏移参数 Phase offset (rad)等于一个固定值 0，

这意味着 OQPSK 频带调制器产生的基带信号的相位偏移 $\theta=\pi/4$。OQPSK 频带调制器模块及其参数设置对话框如图 8-63 所示。

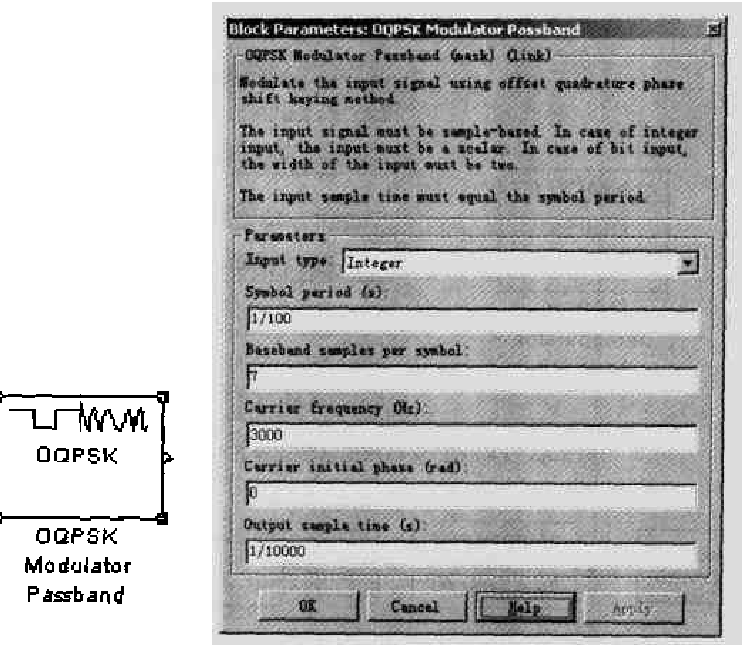



图 8-63　OQPSK 频带调制器模块及其参数设置对话框

OQPSK 频带调制器模块有以下几个参数。

■　Input type（输入信号类型）

当输入信号类型设置为 Integer 时，OQPSK 频带调制器的输入信号是介于 0 和 3 之间的整数；当输入信号类型设置为 Bit 时，OQPSK 频带调制器的输入信号是一个二进制向量，其中每两个二进制位表示一个四进制整数。

■　Symbol period (s)（符号周期）

OQPSK 频带调制器的输入信号（整数或二进制向量）的符号周期（单位：秒）。

■　Baseband samples per symbol（输入符号的基带抽样数）

OQPSK 频带调制器中每个输入符号（整数或二进制向量）产生的基带调制信号的抽样个数。

■　Carrier frequency (Hz)（载波频率）

OQPSK 频带调制器的载波频率（单位：Hz）。

■　Carrier initial phase (rad)（载波初始相位）

OQPSK 频带调制器载波的初始相位（单位：弧度）。

■　Samples per symbol（输出信号采样数）

对应于每一个输入信号 OQPSK 频带调制器产生的输出信号的抽样点的个数。

OQPSK 频带解调器对 OQPSK 频带调制信号进行解调，还原得到调制前的整数序列或二进制序列。OQPSK 频带解调器模块及其参数设置对话框如图 8-64 所示。

OQPSK 频带解调器模块是由 OQPSK 基带解调器模块构成的，它首先把频带信号转换成基带信号，然后通过 OQPSK 基带解调器模块把这个基带信号转换成解调信号。OQPSK 频带解调器模块的内部结构如图 8-65 所示。

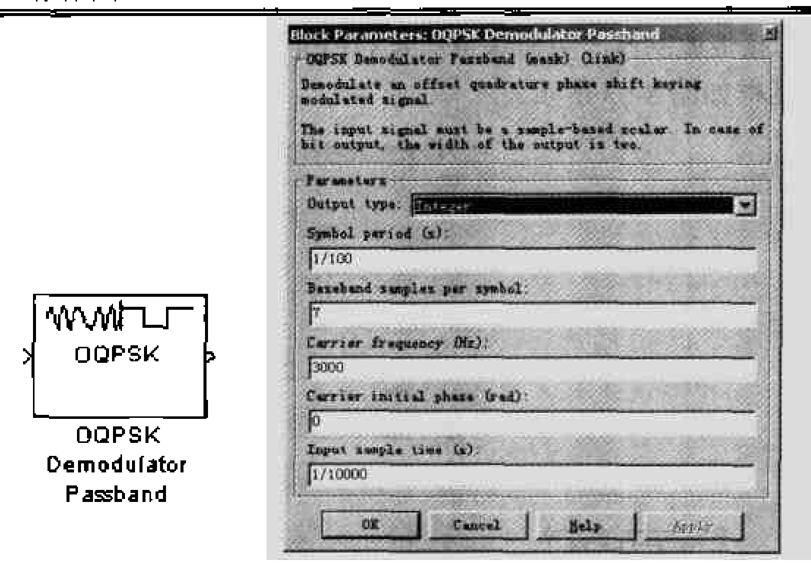



图 8-64　OQPSK 频带解调器模块及其参数设置对话框

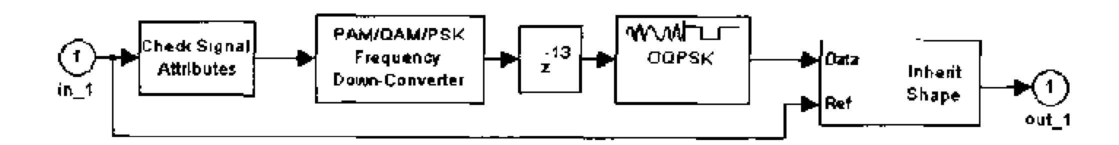



图 8-65　OQPSK 频带解调器模块的内部结构

OQPSK 频带解调器模块主要有以下几个参数。

■　Output type（输出信号类型）

当输出信号类型设置为 Integer 时，OQPSK 频带解调器的输出信号是介于 0 和 3 之间的整数；当输出信号类型设置为 Bit 时，OQPSK 频带解调器的输出信号是一个二进制向量，其中每两个二进制位表示一个四进制整数。

■　Symbol period (s)（符号周期）

OQPSK 频带解调器的输出信号（整数或二进制向量）的符号周期（单位：秒）。

■　Baseband samples per symbol（输出符号的基带抽样数）

OQPSK 频带解调器中与每个输出符号（整数或二进制向量）相对应的基带解调符号的抽样个数。

■　Carrier frequency (Hz)（载波频率）

OQPSK 频带解调器的载波频率（单位：Hz）。

■　Carrier initial phase (rad)（载波初始相位）

OQPSK 频带解调器载波的初始相位（单位：弧度）。

■　Input sample time（输入信号采样数）

OQPSK 频带解调器每个输入信号的抽样点的个数。

### 8.6.9　实例 8.4——OQPSK 在 IS-95 反向信道中的应用

IS-95 前向信道采用了 QPSK 调制方式，而在反向信道中则采用了 OQPSK，其中 Q 支路信号比 I 支路信号落后半个码片周期，以提高调制信号的整形和同步效果。本节将对 QPSK 和 OQPSK 两种调制方式的抗噪声性能进行比较。图 8-66 所示是 OQPSK 调制和解调系统的仿真模型。

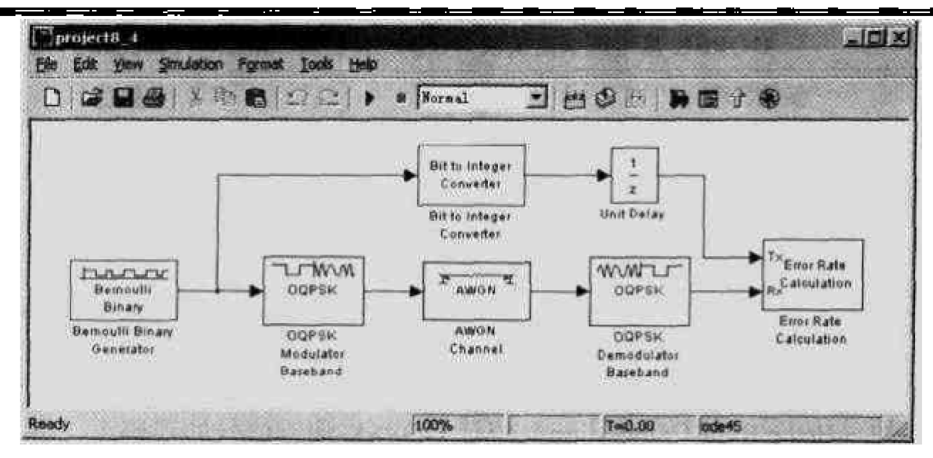



图 8-66　OQPSK 调制的仿真模型

在 OQPSK 调制和解调系统中，贝努利二进制序列产生器（Bernoulli Binary Generator）产生一个长度为 2 的二进制向量，它们分别表示 OQPSK 基带调制器模块（OQPSK Modulator Baseband）两个支路的输入信号。表 8-21 和表 8-22 是这两个模块的参数设置情况。

表 8-21　　贝努利二进制序列产生器（Bernoulli Binary Generator）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | xInitialSeed |
| Sample time | xSampleTime |
| Frame-based outputs | Unchecked |
| Interlpret vector parameters as 1-D | Unchecked |

表 8-22　　OQPSK 基带调制器模块（OQPSK Modulator Baseband）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | OQPSK Modulator Baseband |
| Input type | Bit |
| Phase offset (rad) | xPhaseOffset |
| Samples per symbol | xSamplesPerSymbol |

为了考察 OQPSK 的抗噪声性能，基带调制信号先经过一个加性高斯白噪声模块（AWGN Channel），然后由 OQPSK 基带解调器模块（OQPSK Demodulator Baseband）对其实施解调。表 8-23 和表 8-24 分别列出了这两个模块的参数设置。

表 8-23　　加性高斯白噪声模块（AWGN Channel）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | AWGN Channel |
| Initial seed | 67 |
| Mode | Signal to noise ratio (SNR) |
| SNR (dB) | xSNR |
| Input signal power (watts) | 1 |

表 8-24　　OQPSK 基带解调器模块（OQPSK Demodulator Baseband）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | OQPSK Demodulator Baseband |
| Output type | Integer |
| Phase offset (rad) | xPhaseOffset |
| Samples per symbol | xSamplesPerSymbol |

OQPSK 基带解调器的输出信号是一个四进制整数，为此需要把贝努利二进制序列产生器的输出信号转换成整数，这是通过数值转换模块（Bit to Integer Converter）实现的。此后误码率统计模块（Error Rate Calculation）统计 OQPSK 调制信号的误码率，并且把误码率统计结果保存在工作区变量 xErrorRate 中。数值转换模块和误码率统计模块的参数设置如表 8-25 和表 8-26 所示。

表 8-25　　数值转换模块（Bit to Integer Converter）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bit to Integer Converter |
| Number of bits per integer | 2 |

表 8-26　　误码率统计模块（Error Rate Calculation）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Error Rate Calculation |
| Receive delay | 0 |
| Computation delay | 0 |
| Computation mode | Entire frame |
| Output data | Workspace |
| Variable name | xErrorRate |
| Reset port | Unchecked |
| Stop simulation | Unchecked |

最后编写 M 文件 project8_4main.m，通过这个脚本程序对仿真模型的各个变量进行赋值，然后计算不同信噪比条件下 QPSK 和 OQPSK 的误码率，并且根据仿真的结果绘制曲线图。M 文件 project8_4main.m 的代码如下：

```
% 设置调制信号的抽样间隔
xSampleTime = 1/10000;
% 设置仿真时间的长度
xSimulationTime = 10;
% 设置随机数产生器的初始化种子
xInitialSeed = [61 71];
% 设置 OQPSK 调制的初始相位
xPhaseOffset = pi/4;
% 设置 OQPSK 调制信号的抽样个数
xSamplesPerSymbol = 1;
```

```
% 设置误码率统计模块的接收时延
xReceiveDelay = 0;
% x 表示信噪比的取值范围
x = 0:10;
% y1 表示 OQPSK 调制的误符号率
y1 = x;
% y2 表示 QPSK 调制的误符号率
y2 = x;
for i = 1:length(x)
    % 信噪比依次取向量 x 的数值
    xSNR = x(i);
    % 执行 OQPSK 仿真模型
    sim('project8_4');
    % 从 xErrorRate 中获得调制信号的误码率
    y1(i) = xErrorRate(1);
    % 执行 QPSK 仿真模型
    sim('project8_2');
    % 从 xErrorRate 中获得调制信号的误码率
    y2(i) = xErrorRate(1);
end

% 绘制信噪比与误码率的关系曲线
semilogy(x,y1,'r',x,y2,'b');
```

在 MATLAB 工作区中输入命令行“project8_4”，启动仿真程序运行。这个仿真将在相同的信噪比条件下运行 OQPSK 仿真模型和实例 8.2 中的 QPSK 仿真模型。图 8-67 所示是仿真结束之后得到的两种调制方式的误码率曲线。

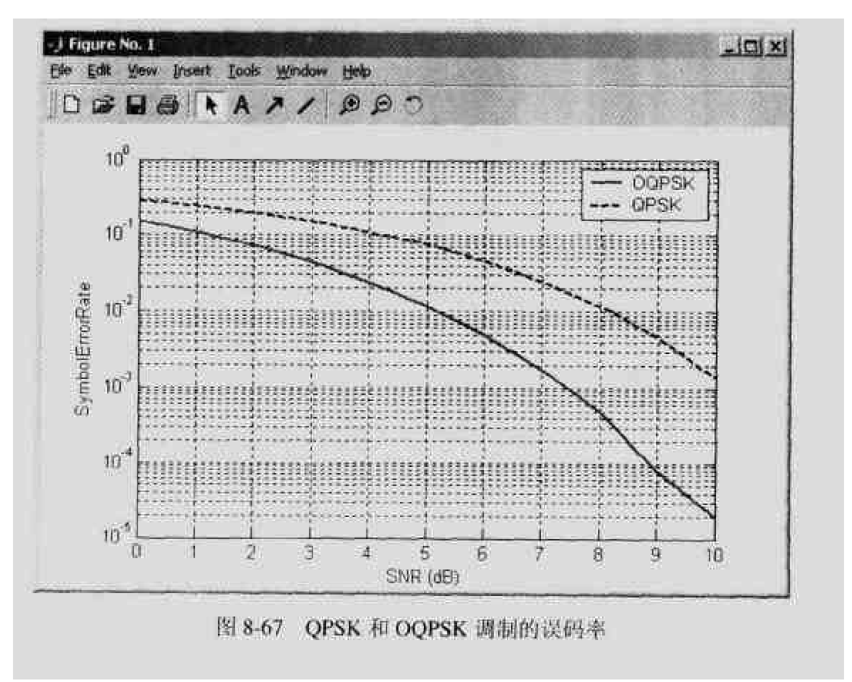



图 8-67　QPSK 和 OQPSK 调制的误码率

图 8-67 所示中的两条曲线分别表示 QPSK 调制和 QQPSK 调制的误码率性能。从图 8-67 所示中可以看到，在相同的数据源和信噪比条件下，OQPSK 调制信号的性能优于 QPSK。例如，当信噪比等于 10 分贝时，QPSK 调制信号的误码率约等于 0.6%，而 OQPSK 调制信号的误码率则只有 0.002%。由于 IS-95 反向信道采用非相干解调，因此采用 OQPSK 而不是 QPSK 能够提高反向信道的接收性能。

## 8.6.10 基带 M-PSK 调制

M 相相移键控（M-PSK，M-ary Phase Shift Keying）是一种多进制数字调制方式，它利用载波的多种不同相位来表示数字信息，较为常用的是四相制和八相制。假设 M 相相移键控的输入信号是 $x(t)$，输出信号是 $y(t)$，则 $x(t)$ 的取值范围是 $\{m \mid m \in Z, 0 \le m < M\}$，且 $y(t) = \exp(j\theta + j2\pi x(t)/M)$，其中 $\theta$ 是 M 相相移键控的相位偏移。因此，M 相相移键控产生的星座图中有 $M$ 个点，这 $M$ 个点位于同一个圆周上，并且把整个圆周分成 $M$ 个相等的间隔。

M-PSK 基带调制器（M-PSK Modulator Baseband）对输入信号实施 M 相相移键控调制，产生复数形式的基带调制信号。M-PSK 基带调制器的输入信号既可以是介于 0 和 $M$-1 之间的整数，也可以是一个长度为 $k$ 的二进制向量，$k = \log_2 M$。M-PSK 基带调制器模块及其参数设置对话框如图 8-68 所示。

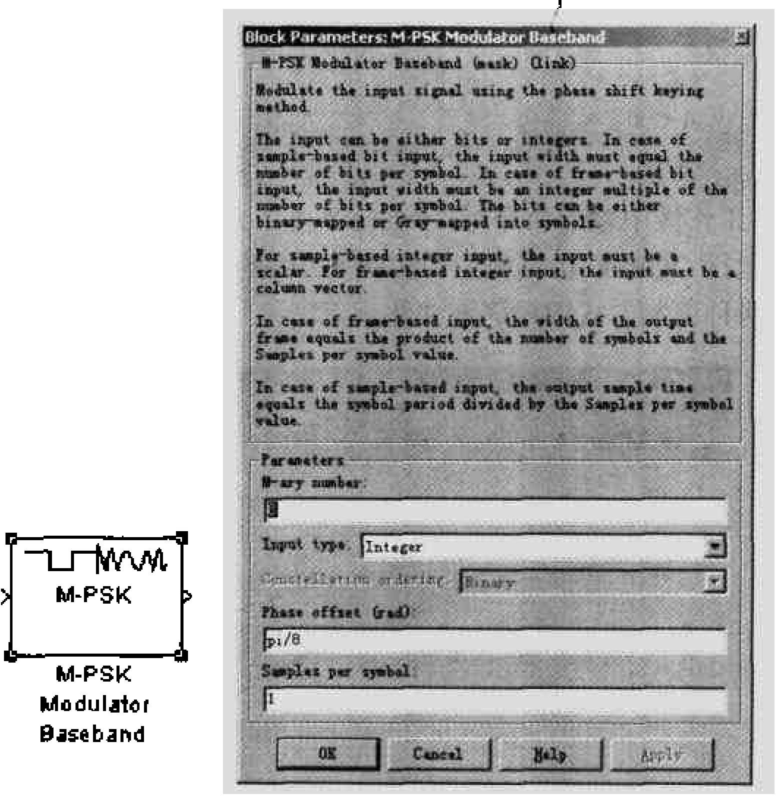



图 8-68 M-PSK 基带调制器模块及其参数设置对话框

M-PSK 基带调制器模块主要有以下几个参数。

- M-ary number（M 相数）

M-PSK 基带调制信号使用的相位的个数。

■　Input type（输入信号类型）

M-PSK 基带调制器输入信号的类型。当本参数设置为 Integer 时，M-PSK 基带调制器的输入信号是介于 0 和 $M$-1 之间的整数；当本参数设置为 Bit 时，M-PSK 基带调制器的输入信号是一个长度为 $k$ 的二进制向量，并且 $k=\log_2 M$。

■　Constellation ordering（星座图编码方式）

当输入信号类型设置为 Bit 时，Constellation ordering 用来指定输入的每 k 个二进制符号与 M-PSK 基带调制星座图中各个点的对应关系：如果星座图编号方式设置为 Binary，MATLAB 把输入的 $k$ 个二进制符号当作一个自然二进制序列；如果星座图编号方式设置为 Gray，则 MATLAB 把输入的 $k$ 个二进制符号当作一个 Gray 码。

■　Phase offset (rad)（相位偏移）

M-PSK 基带调制信号的相位偏移 $\theta$（单位：弧度）。

■　Samples per symbol（输出信号采样数）

对应于每一个输入信号 M-PSK 基带调制器产生的输出信号的抽样点的个数。

M-PSK 基带解调器（M-PSK Demodulator Baseband）对 M 相相移键控调制信号进行解调。假设 M-PSK 基带解调器的输入信号是 $y(t)$，输出信号是 $z(t)$，相位偏移是$\theta$，则当输入信号 $y(t)=\exp(j\theta+j2\pi m/M)$ 时，$z(t)=m$。M-PSK 基带解调器模块及其参数设置对话框如图 8-69 所示。

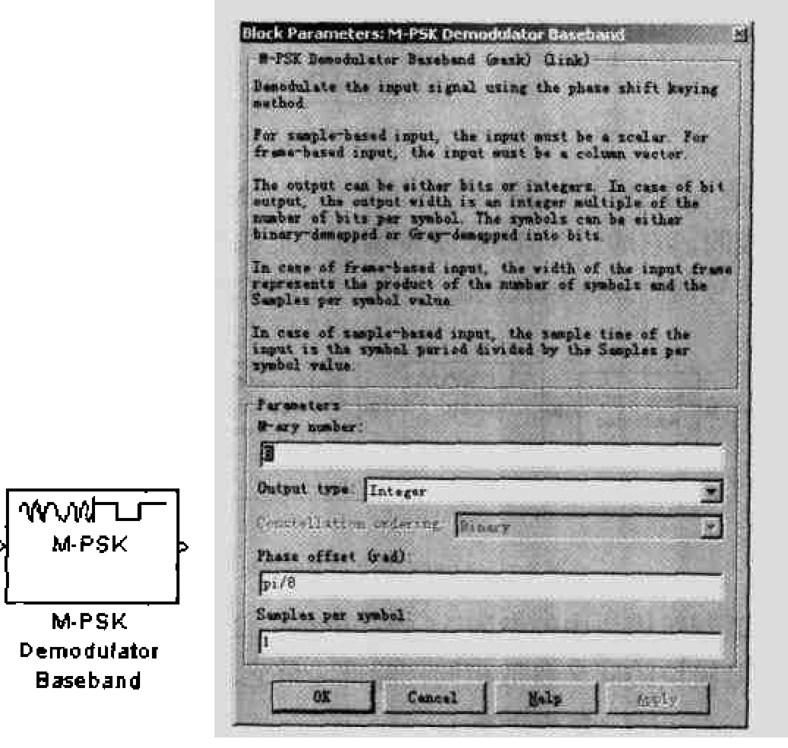



图 8-69　M-PSK 基带解调器模块及其参数设置对话框

M-PSK 基带解调器模块的参数与 M-PSK 基带调制器模块相同。

## 8.6.11　频带 M-PSK 调制

M-PSK 频带调制器（M-PSK Modulator Passband）对输入信号实施 M 相相移键控调制，产生频带调制信号。M-PSK 频带调制器的输入信号既可以是介于 0 和 $M$-1 之间的整数，也

可以是一个长度为 $k$ 的二进制向量，$k=\log_2 M$ 。图 8-70 所示是 M-PSK 频带调制器模块及其参数设置对话框。

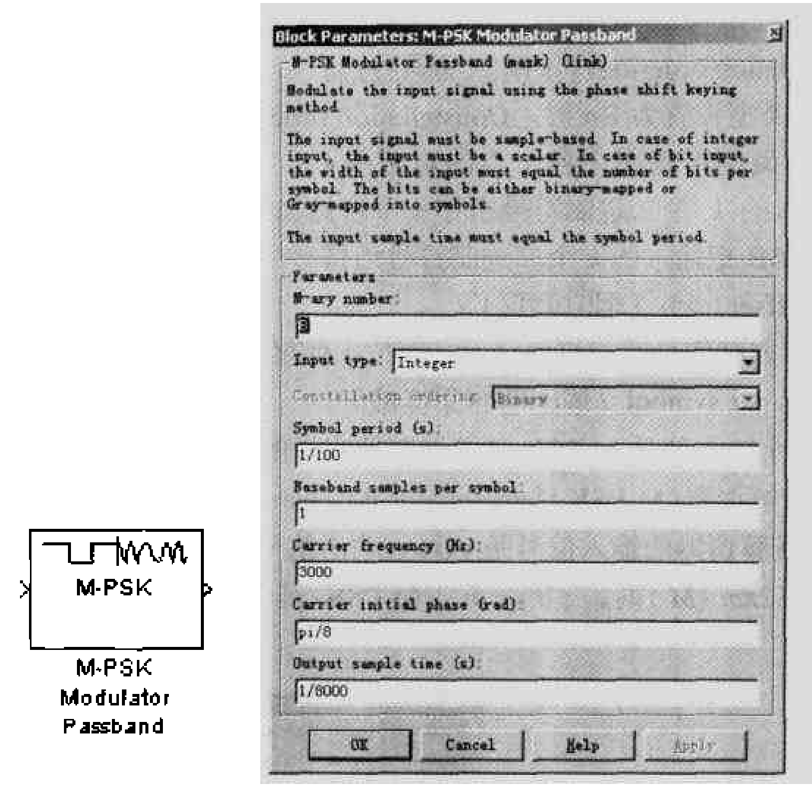



图 8-70　M-PSK 频带调制器模块及其参数设置对话框

图 8-71 所示是 M-PSK 频带调制器的内部结构。M-PSK 频带调制器模块是由 M-PSK 基带调制器模块构成的，它通过 M-PSK 基带调制器把输入信号转换成 M-PSK 基带调制信号，然后把这个基带信号调制到高频载波上，形成 M-PSK 频带调制信号。

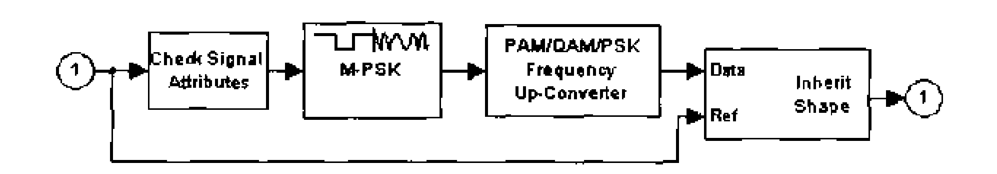



图 8-71　M-PSK 频带调制器模块的内部结构

M-PSK 频带调制器模块中的参数 M-ary number、Input type 以及 Constellation ordering 与 M-PSK 基带调制器模块的同名参数相同，除此之外它还有以下几个参数。

- Symbol period (s)（符号周期）

M-PSK 频带调制器的输入信号（整数或二进制向量）的符号周期（单位：秒）。

- Baseband samples per symbol（输入符号的基带抽样数）

M-PSK 频带调制器中每个输入符号（整数或二进制向量）产生的每个基带调制信号的抽样点的个数。

- Carrier frequency (Hz)（载波频率）

M-PSK 频带调制器的载波频率（单位：Hz）。

■　Carrier initial phase (rad)（载波初始相位）

M-PSK 频带调制器载波的初始相位（单位：弧度）。

■　Samples per symbol（输出信号采样数）

对应于每一个输入信号 M-PSK 频带调制器产生的输出信号的抽样点的个数。

M-PSK 频带解调器（M-PSK Demodulator Passband）对输入的 M-PSK 频带调制信号实施解调。图 8-72 所示是 M-PSK 频带解调器模块及其参数设置对话框。

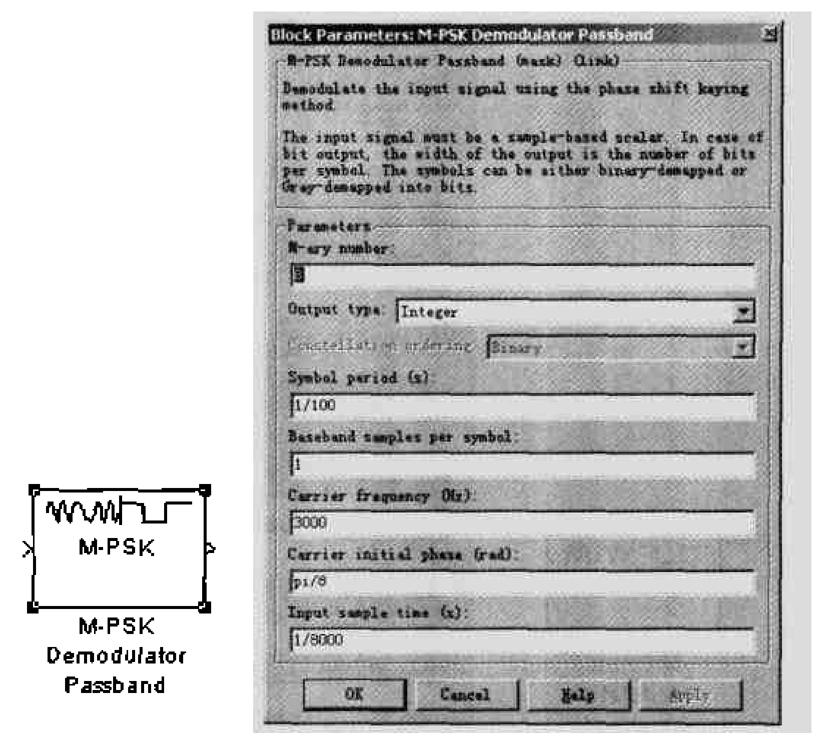



图 8-72　M-PSK 频带解调器模块及其参数设置对话框

M-PSK 频带解调器模块首先把频带调制信号转换成基带调制信号，然后通过 M-PSK 基带解调器对这个基带调制信号进行解调，其内部结构如图 8-73 所示。

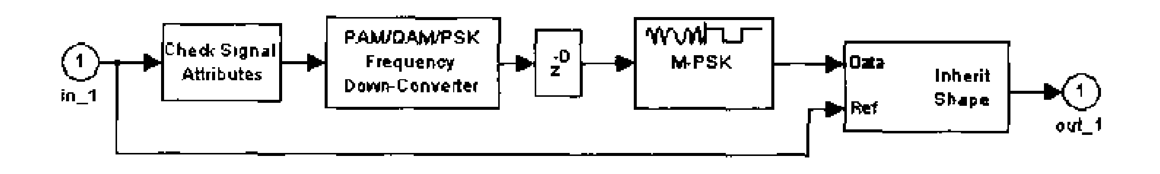



图 8-73　M-PSK 频带解调器模块的内部结构

M-PSK 频带解调器模块的参数与 M-PSK 频带调制器模块相同。

## 8.6.12　基带 M-DPSK 调制

M 相差分相移键控（M-DPSK，M-ary Differential Phase Shift Keying）也是一种多进制数字调制方式，与 M-PSK 不同的是，M-DPSK 利用载波的不同相位差来表示数字信息。假设 M 相相移键控的输入信号是 $x(t)$，输出信号是 $y(t)$，则 $x(t)$ 的取值范围是 $\{m \mid m \in Z, 0 \le m < M\}$。对于第一个输入信号 $x(1)$，$y(1) = \exp(j\theta + j2\pi x(1)/M)$；对于其

后的输入信号 $x(t)$， $y(t)=y(t-1)\exp(j\theta+j2\pi x(t)/M)$，其中$\theta$是 M 相差分相移键控的相位偏移。

M-DPSK 基带调制器（M-DPSK Modulator Baseband）对输入信号实施 M 相差分相移键控调制，产生复数形式的基带调制信号。M-DPSK 基带调制器的输入信号既可以是介于 0 和 $M$-1 之间的整数，也可以是一个长度为 $k$ 的二进制向量，其中 $k=\log_2 M$ 。M-DPSK 基带调制器模块及其参数设置对话框如图 8-74 所示。

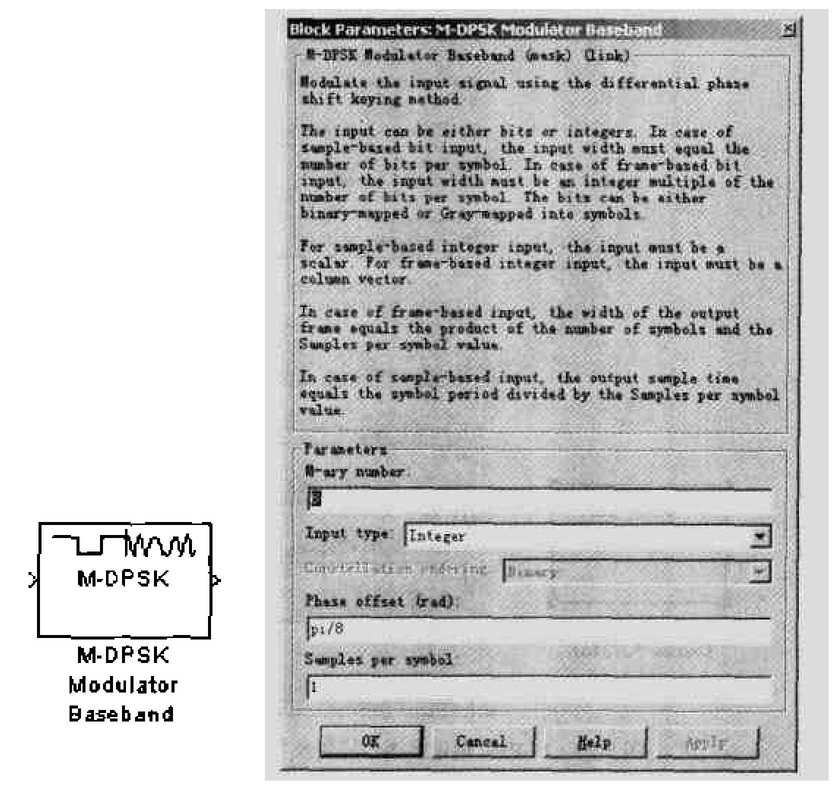



图 8-74 M-DPSK 基带调制器模块及其参数设置对话框

M-DPSK 基带调制器模块主要有以下几个参数。

■ M-ary number（M 相数）

M-DPSK 基带调制信号使用的相位差的个数。

■ Input type（输入信号类型）

M-DPSK 基带调制器输入信号的类型。当本参数设置为 Integer 时，M-DPSK 基带调制器的输入信号是介于 0 和 $M$-1 之间的整数；当本参数设置为 Bit 时，M-DPSK 基带调制器的输入信号是一个长度为 $k$ 的二进制向量，并且 $k=\log_2 M$ 。

■ Constellation ordering（星座图编码方式）

当输入信号类型设置为 Bit 时，Constellation ordering 用来指定输入的每 k 个二进制符号与 M-DPSK 基带调制星座图中各个点的对应关系：如果星座图编号方式设置为 Binary，MATLAB 把输入的 $k$ 个二进制符号当作一个自然二进制序列；如果星座图编号方式设置为 Gray，则 MATLAB 把输入的 $k$ 个二进制符号当作一个 Gray 码。

■ Phase offset (rad)（相位偏移）

M-DPSK 基带调制信号的相位偏移$\theta$（单位：弧度）。

- Samples per symbol（输出信号采样数）

对应于每一个输入信号 M-DPSK 基带调制器产生的输出信号的抽样点的个数。

M-DPSK 基带解调器（M-DPSK Demodulator Basceband）对 M 相相移键控调制信号进行解调。假设 M-DPSK 基带解调器的输入信号是 $y(t)$，输出信号是 $z(t)$，相位偏移是 $\theta$。对于第一个输入信号 $y(1)$，如果 $y(1)=\exp(j\theta+j2\pi m/M)$，则 $z(1)=m$；对于其后的输入信号 $y(t)$，如果 $y(t)/y(t-1)=\exp(j\theta+j2\pi m/M)$，则 $z(t)=m$。M-DPSK 基带解调器模块及其参数设置对话框如图 8-75 所示。

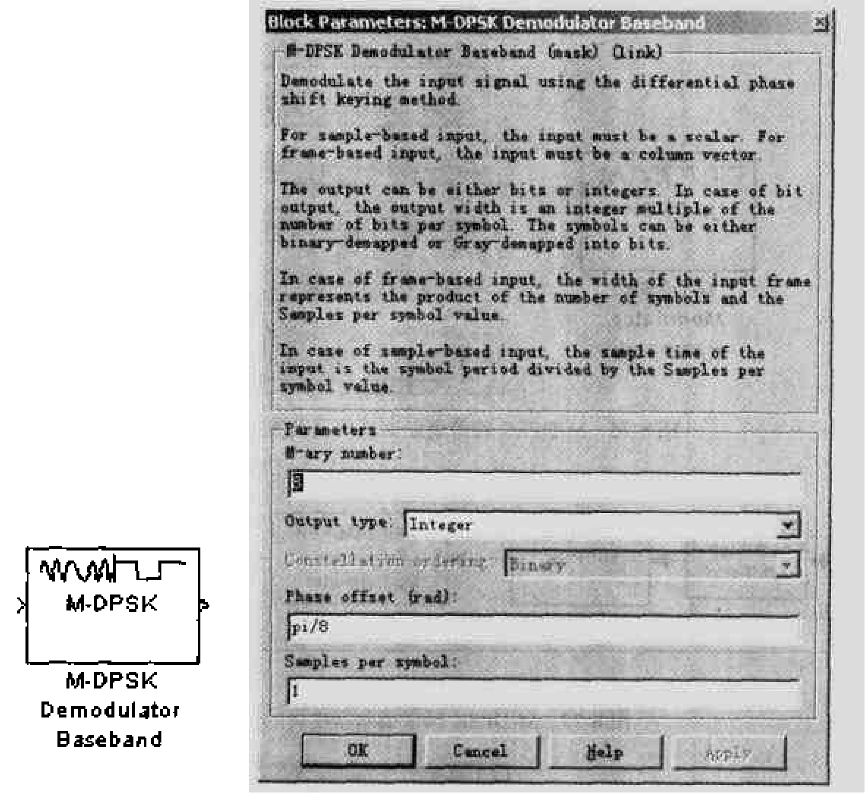



图 8-75　M-DPSK 基带解调器模块及其参数设置对话框

M-DPSK 基带解调器模块的参数与 M-DPSK 基带调制器模块相同。

## 8.6.13　频带 M-DPSK 调制

M-DPSK 频带调制器（M-DPSK Modulator Passband）对输入信号实施 M 相差分相移键控调制，产生频带调制信号。图 8-76 所示是 M-DPSK 频带调制器模块及其参数设置对话框。

M-DPSK 频带调制器模块是由 M-DPSK 基带调制器模块构成的，它通过 M-DPSK 基带调制器把输入信号转换成 M-DPSK 基带调制信号，然后把这个基带信号调制到高频载波上，形成 M-DPSK 频带调制信号。图 8-77 所示是 M-DPSK 频带调制器的内部结构。

M-DPSK 频带解调器（M-DPSK Demodulator Passband）对输入的 M-DPSK 频带调制信号实施解调。图 8-78 所示是 M-DPSK 频带解调器模块及其参数设置对话框。

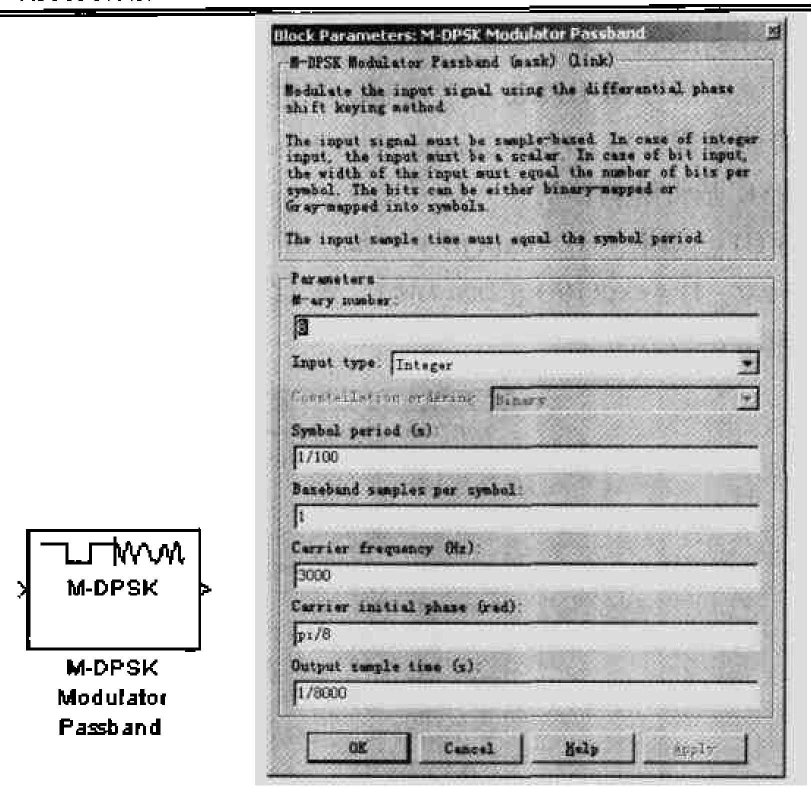



图 8-76　M-DPSK 频带调制器模块及其参数设置对话框

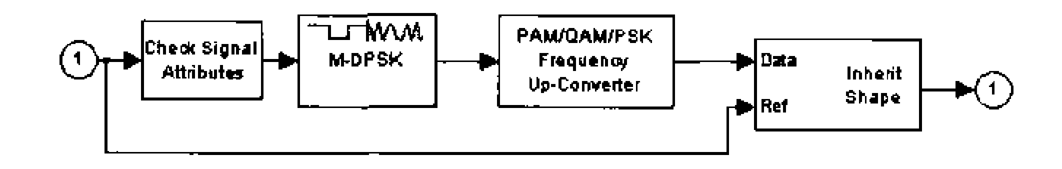



图 8-77　M-DPSK 频带调制器模块的内部结构

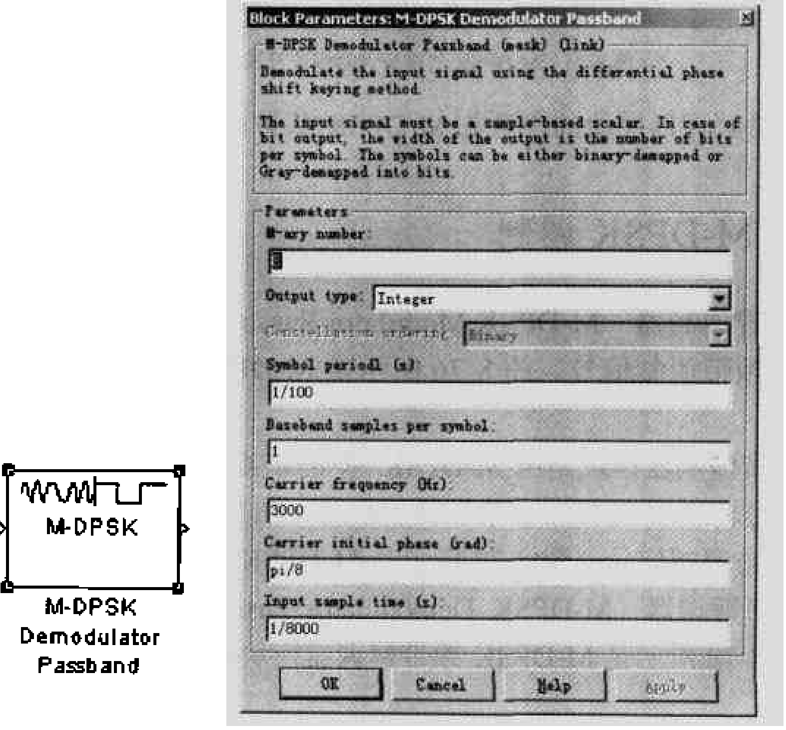



图 8-78　M-PSK 频带解调器模块及其参数设置对话框

M-DPSK 频带解调器模块首先把频带调制信号转换成基带调制信号，然后通过 M-DPSK 基带解调器对这个基带调制信号进行解调，其内部结构如图 8-79 所示。

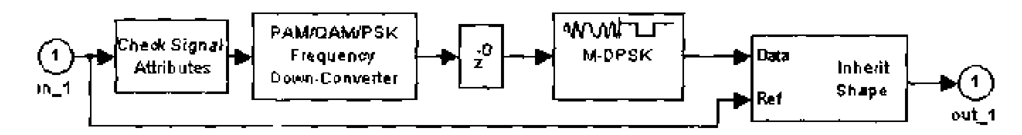



图 8-79　M-DPSK 频带解调器模块的内部结构

M-DPSK 频带调制器模块的参数与 M-DPSK 频带调制器模块相同。

# 8.7　数字连续相位调制

连续相位调制（CPM，Continuous Phase Modulation）是指通过脉冲整形来平滑调制信号之间的相位转换的调制方式。在 MATLAB 中，连续相位调制可以使用多种形式的脉冲整形滤波器，如矩形滤波器、Raised Cosine 滤波器、Gaussian 滤波器和 Tamed FM 滤波器。

## 8.7.1　基带 CPM 调制

CPM 基带调制器（CPM Modulator Baseband）对输入信号实施连续相位调制，产生基带调制复信号。CPM 基带调制器模块的输入信号可以是整数，也可以是长度为 $\log_2 M$ 的二进制向量，其模块框图和参数设置对话框如图 8-80 所示。

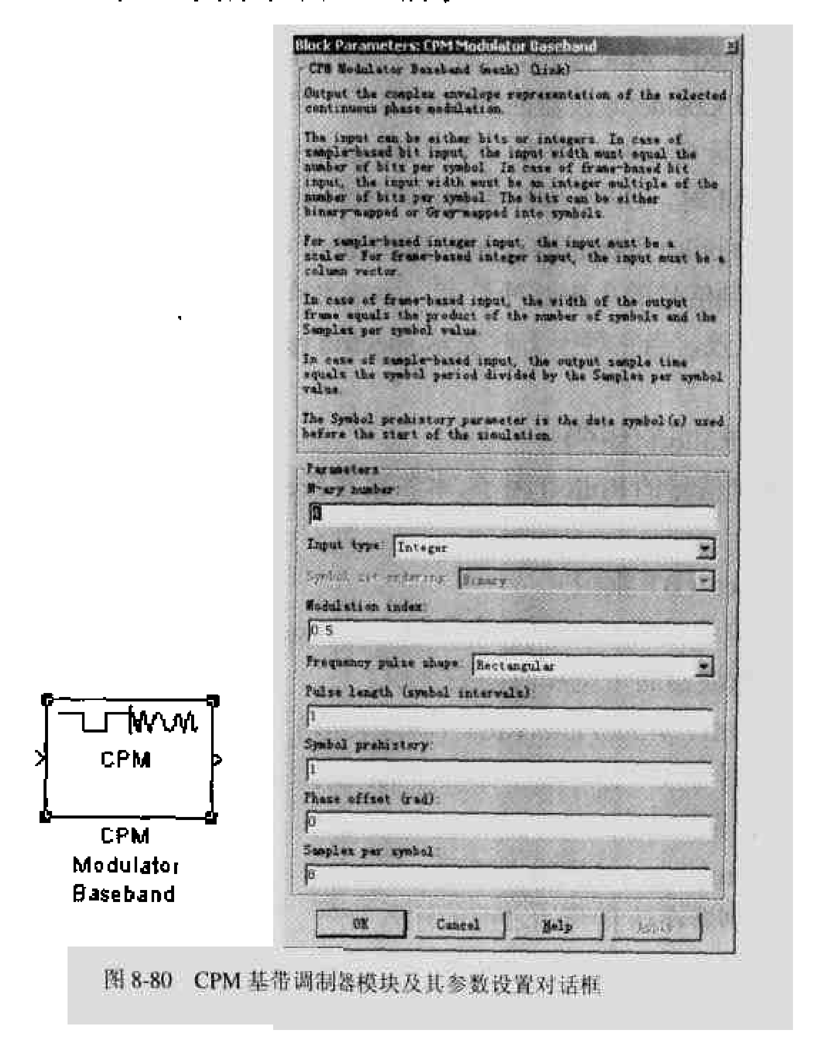



图 8-80　CPM 基带调制器模块及其参数设置对话框

CPM 基带调制器模块主要有以下几个参数。

■ M-ary number（M 相数）

CPM 基带调制器的输入信号的相数。当本参数设置为 $M$ 时，CPM 基带调制器的输入信号是介于 0 和 $M$-1 之间的整数（或相应的二进制向量），并且 $M$ 满足条件 $M=2^k$ 。

■ Input type（输入信号的类型）

CPM 基带调制器输入信号的类型。当本参数设置为 Integer 时，CPM 基带调制器输入信号是一个介于 0 和 $M$-1 之间的整数；当本参数设置为 Bit 时，CPM 基带调制器输入信号是一个长度为 $\log_2 M$ 的二进制向量。

■ Symbol set ordering（输入符号编码方式）

CPM 基带调制器输入的二进制信号的编码方式，本参数在 Input type 设置为 Bit 时有效。当本参数设置为 Binary 时，输入信号是按照自然二进制码进行编码的信号；当本参数设置为 Gray 时，输入信号是按照 Gray 码进行编码的信号。

■ Modulation index（调制指数）

CPM 基带调制信号的调制指数 m。当输入信号等于 1（整数）时 CPM 基带调制信号的相位偏移就等于 $m\pi$ 。

■ Frequency pulse shape（频率脉冲整形方式）

CPM 基带调制器的脉冲整形方式。当本参数设置为 Rectangular 时，CPM 基带调制器对调制信号实施矩形整形；当本参数设置为 Raised Cosine 时，CPM 基带调制器对调制信号实施 Raised Cosine 整形；当本参数设置为 Spectral Raised Cosine 时，CPM 基带调制器对调制信号实施 Spectral Raised Cosine 整形，这时候参数 Main lobe pulse duration 表示主瓣脉冲的长度，Rolloff 表示 Raised Cosine 滤波器的 rolloff 因子；当本参数设置为 Gaussian 时，CPM 基带调制器对调制信号实施 Gaussian 整形，参数 BT product 表示滤波器带宽和时间的乘积；当本参数设置为 Tamed FM 时，CPM 基带调制器对调制信号实施 Tamed FM 整形。

■ Main lobe pulse duration (symbol intervals)（主瓣脉冲长度）

CPM 基带调制信号的主瓣脉冲长度。本参数在 Frequency pulse shape 设置为 Spectral Raised Cosine 时有效，当它等于 $k$ 时表示 spectral raised cosine 脉冲的主瓣长度是输入符号周期的 $k$ 倍。

■ Rolloff（Rolloff 因子）

CPM 基带调制信号的 Rolloff 因子。本参数在 Frequency pulse shape 设置为 Spectral Raised Cosine 时有效。

■ BT product（BT 乘积）

CPM 基带调制信号的 BT 乘积。本参数在 Frequency pulse shape 设置为 Gaussian 时有效，它表示 Gaussian 滤波器的带宽和时间的乘积。

■ Pulse length (symbol intervals)（脉冲长度）

CPM 基带调制器频率脉冲整形滤波器的周期。当本参数等于 $k$ 时，脉冲整形滤波器的周期等于输入符号周期的 $k$ 倍。

■ Symbol prehistory（预输入符号）

CPM 基带调制器在仿真开始之前的输入符号。如果本参数是一个向量，则该向量的长度等于参数 Pulse length 的数值减 1。

■　Phase offset (rad)（相位偏移）

CPM 基带调制信号的初始相位（单位：弧度）。

■　Samples per symbol（输出信号采样数）

对应于每一个输入符号（整数或二进制向量）CPM 基带调制器产生的输出信号的抽样点的个数。

CPM 基带解调器模块对输入的 CPM 基带调制信号进行解调，输出 M 进制信息序列。这个 M 进制输出信号可以是整数，也可以是长度为 $\log_2 M$ 的二进制向量。CPM 基带解调器模块及其参数设置对话框及其参数设置对话框如图 8-81 所示。

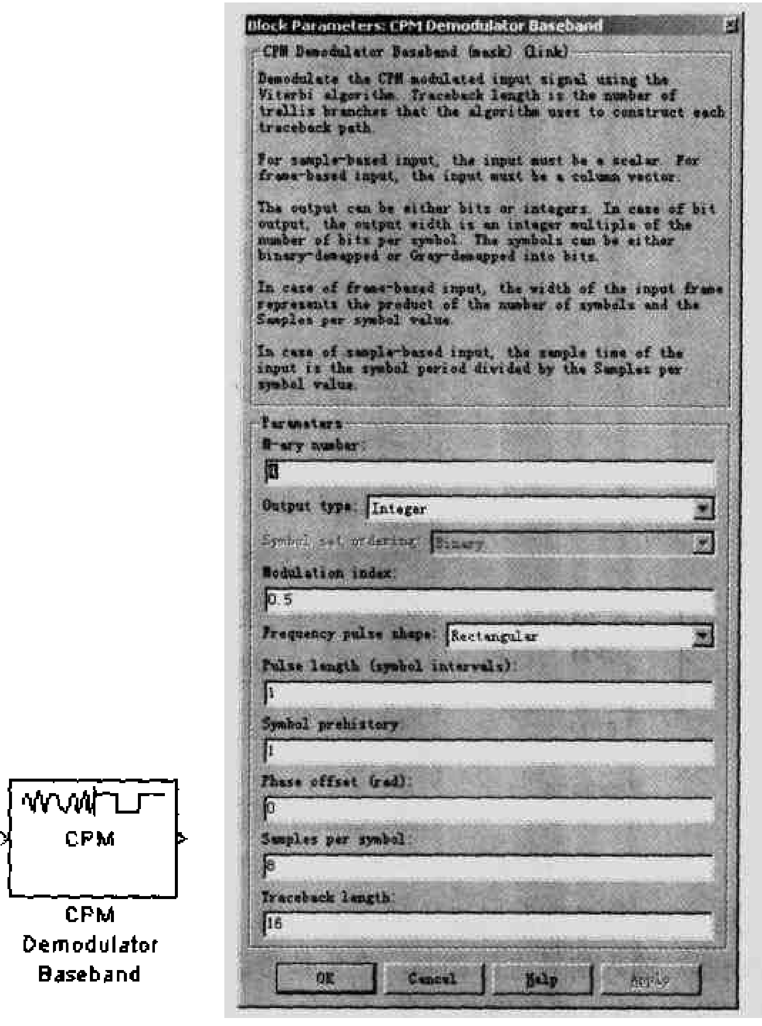



图 8-81　CPM 基带解调器模块及其参数设置对话框

CPM 基带解调器模块采用 Viterbi 算法对 CPM 调制信号进行解调，其中回溯长度 $D$ 表示 Viterbi 算法中使用的解码支路的个数。回溯长度 $D$ 在解码过程中引入了时延，即在 CPM 基带解调器模块产生第一个解码信号之前输出的 0 的个数。当 CPM 基带解调器模块的输入信号是抽样形式的信号时，解码时延等于 $D+1$；当 CPM 基带解调器模块的输入信号是帧格式的信号时，解码时延等于 $D$。

CPM 基带解调器模块除了具有 CPM 基带调制器模块的各个参数之外，还有如下参数：

■　Traceback length（回溯长度）

CPM 基带解调器在 Viterbi 译码过程中的回溯长度 $D$。

## 8.7.2 频带 CPM 调制

CPM 频带调制器模块对 $M$ 进制信息序列进行 CPM 调制，产生 CPM 频带调制信号。CPM 频带调制器模块的输入信号可以是整数，也可以是长度为 $\log_2 M$ 的二进制向量，它产生的输出信号是一个实信号，其模块框图和参数设置对话框如图 8-82 所示。

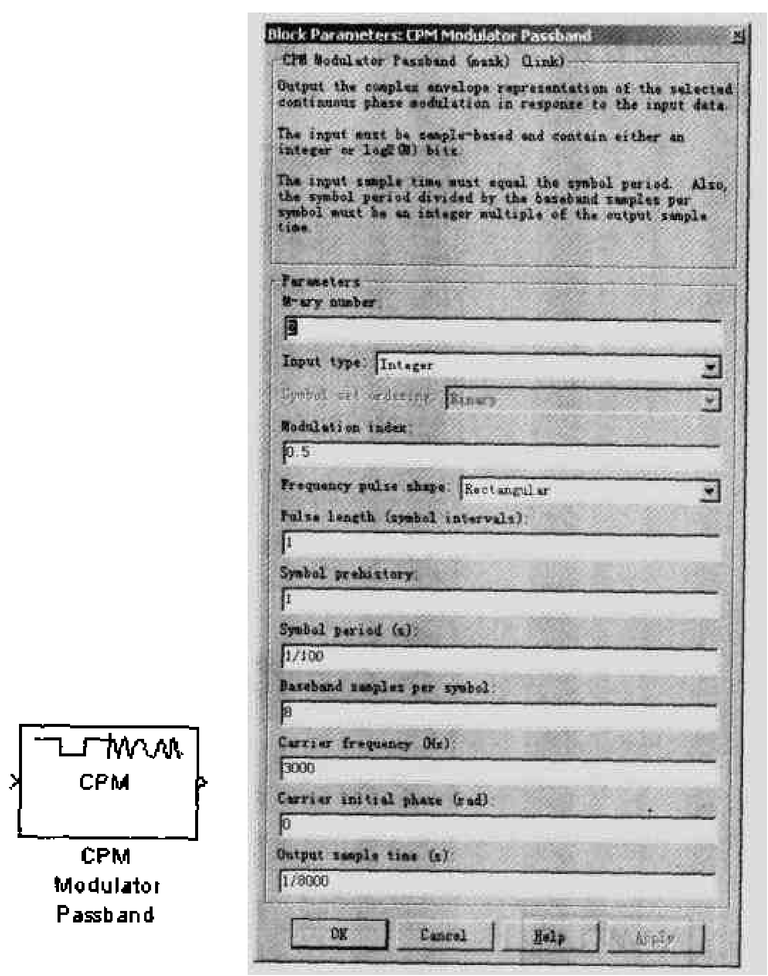



图 8-82　CPM 频带调制器模块及其参数设置对话框

图 8-83 所示是 CPM 频带调制器模块的内部结构。从图中可以看到，CPM 频带调制器使用 CPM 基带调制器对输入信号进行调制，产生基带调制信号，然后把这个调制信号调制到高频载波上，从而得到 CPM 频带调制信号。

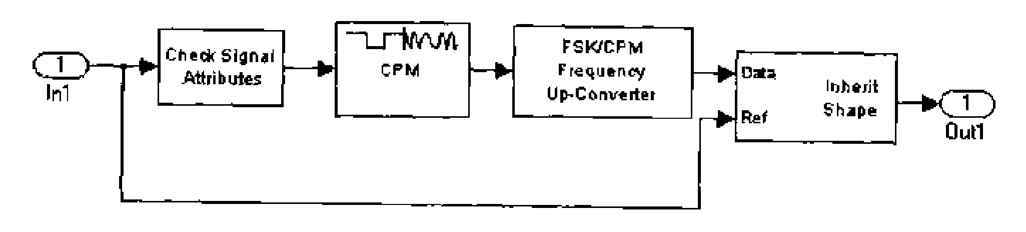



图 8-83　CPM 频带调制器模块的内部结构

CPM 频带调制器模块中的参数 M-ary number、Input type、Symbol set ordering、Modulation index、Frequency pulse shape、Rolloff、BT product、Pulse length 以及 Symbol prehistory 都与 CPM

基带调制器模块中的同名参数相同。除此之外，CPM 频带调制器模块还有以下几个参数。

■　Symbol period (s)（符号周期）

CPM 频带调制器的输入信号（整数或二进制向量）的符号周期（单位：秒）。

■　Baseband samples per symbol（输入符号的基带抽样数）

CPM 频带调制器中每个输入符号（整数或二进制向量）产生的基带调制信号的抽样点的个数。

■　Carrier frequency (Hz)（载波频率）

CPM 频带调制器的载波频率（单位：Hz）。

■　Carrier initial phase (rad)（载波初始相位）

CPM 频带调制器载波的初始相位（单位：弧度）。

■　Samples per symbol（输出信号采样数）

对应于每一个输入信号 CPM 频带调制器产生的输出信号的抽样点的个数。

CPM 频带解调器对高频的 CPM 调制信号进行解调，产生 M 进制的信息序列，这个信息序列中的元素可以是整数，也可以是长度为 $\log_2 M$ 的二进制向量。CPM 频带调制器的输入输出信号都是实信号，其模块框图和参数设置对话框如图 8-84 所示。

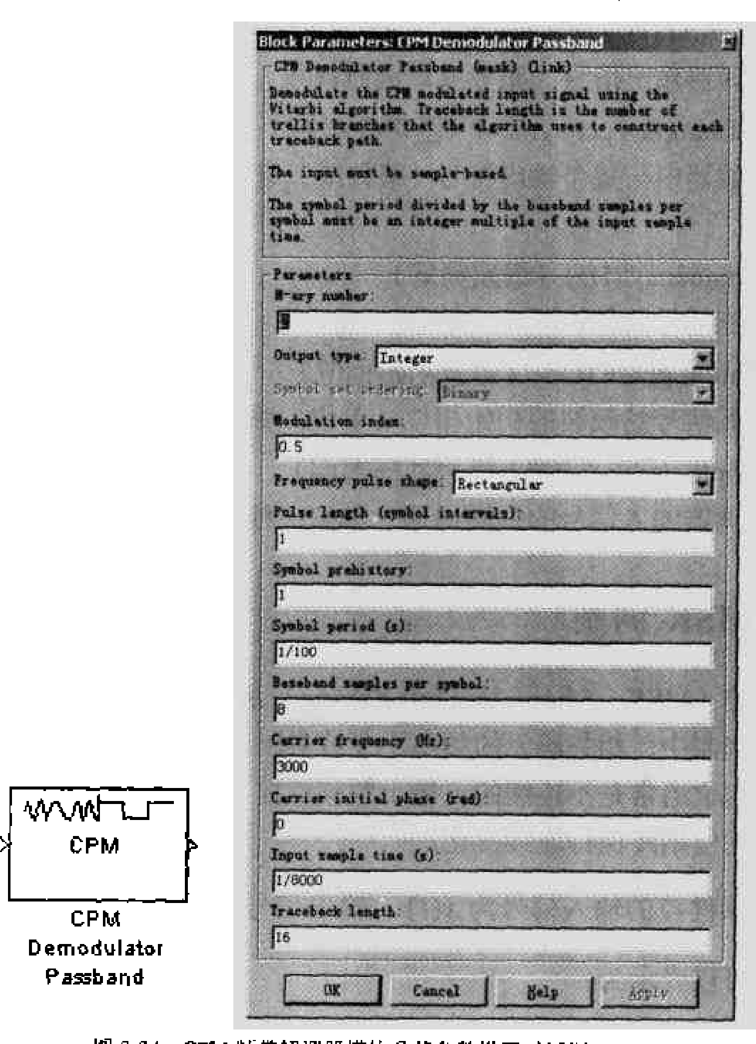



图 8-84　CPM 频带解调器模块及其参数设置对话框

CPM 频带解调器模块把输入的频带调制信号转换成基带调制信号，然后通过 CPM 基带

解调器对这个基带调制信号进行解调，得到与输入信号相对应的解调信号。图 8-85 所示是 CPM 频带解调器模块的内部结构。

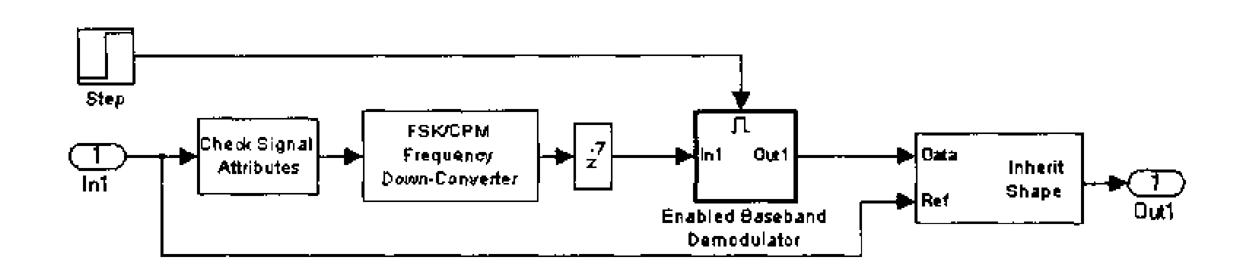



图 8-85　CPM 频带解调器模块的内部结构

CPM 频带解调器模块中的参数 M-ary number、Output type、Symbol set ordering、Modulation index、Frequency pulse shape、Rolloff、BT product、Pulse length、Symbol prehistory 以及 Traceback length 与 CPM 频带解调器模块中的同名参数相同，另外，它还有以下几个参数。

- Symbol period (s)（符号周期）

CPM 频带解调器输出信号（整数或二进制向量）的符号周期（单位：秒）。

- Baseband samples per symbol（基带抽样数）

CPM 频带解调器中与每个输出符号（整数或二进制向量）相对应的基带调制信号的抽样点的个数。

- Carrier frequency (Hz)（载波频率）

CPM 频带调制信号的载波频率（单位：Hz）。

- Carrier initial phase (rad)（载波初始相位）

CPM 频带调制信号的载波的初始相位（单位：弧度）。

- Input sample time(s)（输入信号抽样周期）

CPM 频带解调器输入信号的抽样周期（单位：秒）。

## 8.7.3　基带 MSK 调制

最小频移键控（MSK，Minimum Shift Keying）是二进制频移键控（BFSK）的一种改进形式。在 BFSK 调制中，相邻码元的频率不变或者跳变一个固定值；在两个相邻的频率跳变的码元之间，其相位通常是不连续的。最小频移键控 MSK 对 BFSK 调制作出了一些改进，使得调制信号的相位始终保持连续变化。

假设最小频移键控的输入信号为 $x(t)$，输出信号为 $y(t)$，输入信号的取值范围是 ±1，则输出信号 $y(t)$ 可以表示为：

$$y(t)=\cos(2\pi f_c t+\frac{\pi x(t)}{2T_s}t+\theta_k)\quad (k-1)T_s\le t\le kT_s \tag{8.5}$$

其中，$f_c$ 是 MSK 调制信号的载波频率，$T_s$ 是输入符号的周期，$\theta_k$ 是第 $k$ 个码元的相位常数，它在时间 $(k-1)T_s \le t \le kT_s$ 内保持恒定。从公式中可以看到，MSK 调制信号中的两个频率分别为：

$$f_1 = f_c + \frac{1}{4T_s}，\ f_2 = f_c - \frac{1}{4T_s}。$$

为了使 MSK 调制信号的相位保持连续，载波频率需满足条件：

$$f_c = \frac{n}{4T_s} = (N + \frac{m}{4})\frac{1}{T_s}，$$

其中 $n$，$N$ 为正整数，$m = 0,1,2,3$。

MSK 调制信号的相位常数 $\theta_k$ 还应该满足相位约束条件：

$$\theta_k = \theta_{k-1} + \frac{(k-1)\pi}{2}(a_{k-1} - a_k) = \begin{cases} \theta_{k-1} & 当a_{k-1} = a_k时 \\ \theta_{k-1} \pm (k-1)\pi & 当a_{k-1} \ne a_k时 \end{cases} \tag{8.6}$$

其中 $a_k$ 表示输入信号 $x(t)$ 在时间 $(k-1)T_s \le t \le kT_s$ 内的数值。这时候 MSK 调制信号在第 $k$ 个码元时刻的相位常数不仅与当前的输入信号 $a_k$ 有关，而且与前面的输入信号 $a_{k-1}$ 及其相位常数 $\theta_{k-1}$ 有关，前后码元之间存在着相关性。MSK 调制信号的优点在于它的功率谱比 BFSK 调制信号紧密，因而比较适合于在窄带信道中传输，对相邻信道的干扰也比较小，具有较好的抗干扰性能。

MSK 基带调制器模块（MSK Modulator Baseband）对输入的双极性信号或二进制信号实施最小频移键控调制，产生基带调制信号，其模块框图和参数设置对话框如图 8-86 所示。

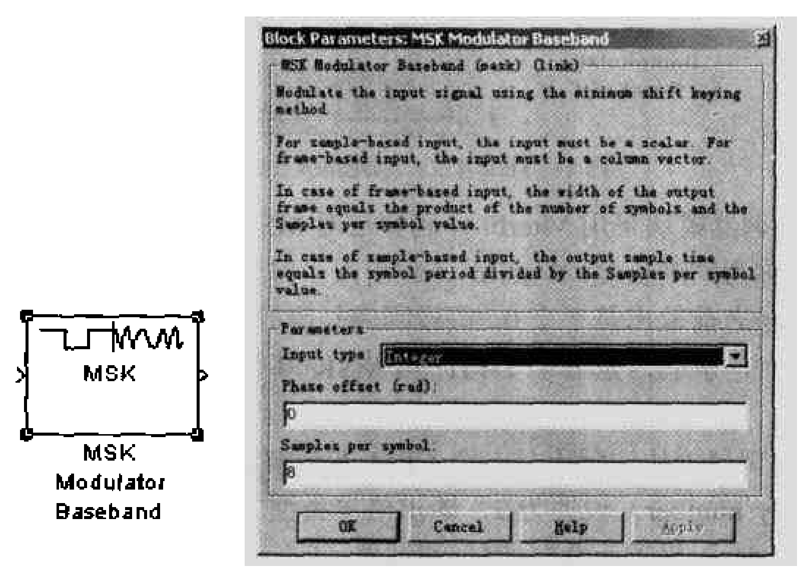



图 8-86　MSK 基带调制器模块及其参数设置对话框

MSK 基带调制器模块是由 CPM 基带调制器构造而来的，其中，CPM 基带调制器模块的 M 相数（M-ary number）设置为 2，输入信号类型（Input type）设置为 Integer，调制指数（Modulation index）设置为 0.5，整形滤波器（Frequency pulse shape）设置为 Rectangular，

脉冲长度（Pulse length）设置为 1，预输入信号（Symbol prehistory）等于 1。

MSK 基带调制器模块有以下 3 个参数。

■ Input type（输入信号类型）

MSK 基带调制器模块输入信号的类型。当本参数设置为 Integer 时，MSK 基带调制器模块的输入信号是一个双极性信号（±1）；当本参数设置为 Bit 时，MSK 基带调制器模块的输入信号是二进制信号（0 或 1）。

■ Phase offset (rad)（相位偏移）

MSK 基带调制器模块输出信号的初始相位（单位：弧度）。

■ Samples per symbol（输出信号抽样数）

对应于每个输入信号 MSK 基带调制器模块产生的输出信号的抽样数。

MSK 基带解调器（MSK Demodulator Baseband）对 MSK 基带调制信号进行解调，得到二进制信息序列（或双极性信号序列）。MSK 基带解调器模块及其参数设置对话框如图 8-87 所示。

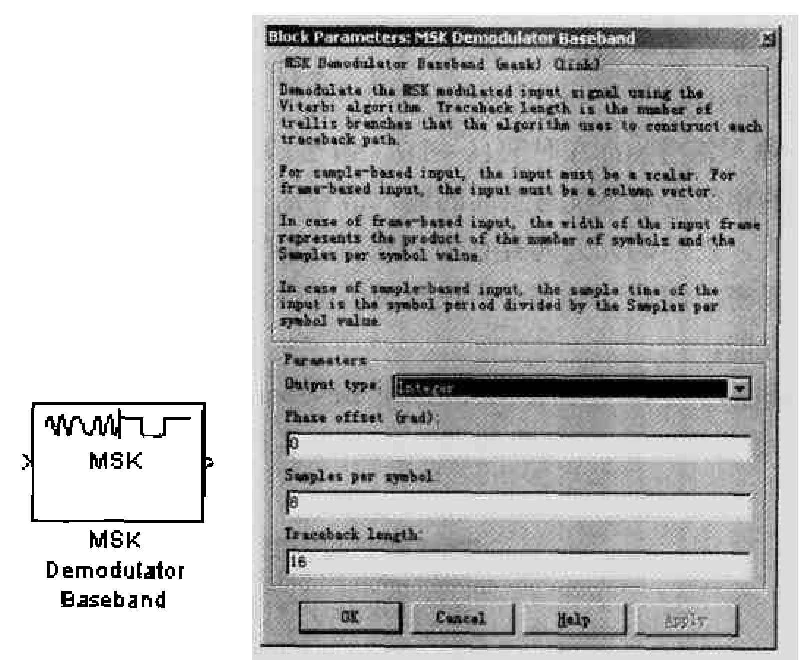



图 8-87 MSK 基带解调器模块及其参数设置对话框

MSK 基带解调器模块采用 Viterbi 译码算法对 MSK 调制信号进行解调，该算法中使用的解码支路的个数称为回溯长度。Viterbi 译码引入了时延，它导致 MSK 基带解调器模块在产生第一个解码信号之前输出若干个 0。假设回溯长度等于 $D$，当 MSK 基带解调器模块的输入信号是抽样信号时，解码时延等于 $D$+1；当 MSK 基带解调器模块的输入信号是帧信号时，解码时延等于 $D$。

MSK 基带解调器模块实际上是一个 CPM 基带解调器模块，它把 CPM 基带解调器模块的 M 相数（M-ary number）设置为 2，输出信号类型（Output type）设置为 Integer，调制指数（Modulation index）设置为 0.5，整形滤波器（Frequency pulse shape）设置为 Rectangular，脉冲长度（Pulse length）设置为 1，预输入信号（Symbol prehistory）等于 1。

MSK 基带解调器模块主要有以下几个参数。

■ Output type（输出信号类型）

MSK 基带解调器模块输出信号的类型。当本参数设置为 Integer 时，MSK 基带解调器模

块的输出信号是一个双极性信号（±1）；当本参数设置为 Bit 时，MSK 基带解调器模块的输出信号是二进制信号(0 或 1)。本参数应该与输入信号相对应的 MSK 基带调制器的 Input type 参数保持一致。

■　Phase offset (rad)（相位偏移）

MSK 基带解调器模块输入信号的初始相位（单位：弧度）。

■　Samples per symbol（输入信号抽样数）

MSK 基带解调器模块每个输入的调制信号的抽样个数。

■　Traceback length（回溯长度）

MSK 基带解调器模块中的 Viterbi 译码器的回溯长度。

## 8.7.4　频带 MSK 调制

MSK 频带调制器模块（MSK Modulator Passband）对输入的双极性信号或二进制信号实施最小频移键控调制，产生频带调制信号，其模块框图和参数设置对话框如图 8-88 所示。

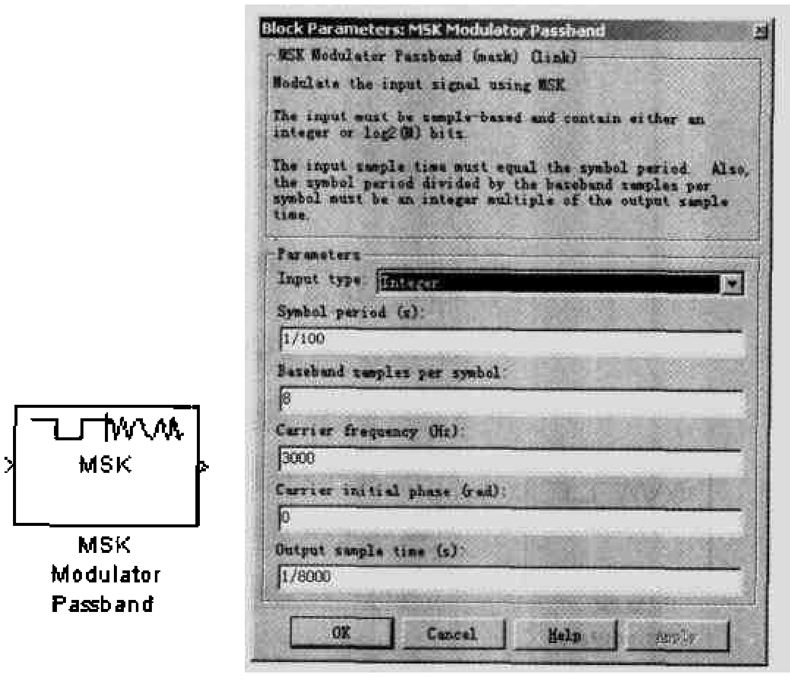



图 8-88　MSK 频带调制器模块及其参数设置对话框

MSK 频带调制器模块首先把输入信号通过一个 MSK 基带调制器，产生基带调制信号，然后把这个基带信号调制到高频载波上，形成频带 MSK 调制信号。实际上 MSK 频带调制器模块是由 CPM 频带调制器构造而来的。在这个 CPM 频带调制器中，M 相数（M-ary number）设置为 2，输入信号类型（Input type）设置为 Integer，调制指数（Modulation index）设置为 0.5，整形滤波器（Frequency pulse shape）设置为 Rectangular，脉冲长度（Pulse length）设置为 1，预输入信号（Symbol prehistory）等于 1。

MSK 频带调制器模块主要有以下几个参数。

■　Input type（输入信号类型）

MSK 频带调制器模块输入信号的类型。当本参数设置为 Integer 时，MSK 频带调制器模块的输入信号是一个双极性信号（±1）；当本参数设置为 Bit 时，MSK 频带调制器模块的输

入信号是二进制信号（0 或 1）。

■ Symbol period (s)（输入符号周期）

MSK 频带调制器模块每个输入符号的周期（单位：秒）。

■ Baseband samples per symbol（输入符号的基带采样数）

MSK 频带调制器模块每个输入符号产生的基带调制信号的采样个数。

■ Carrier frequency (Hz)（载波频率）

MSK 频带调制信号的载波频率（单位：Hz）。

■ Carrier initial phase (rad)（载波相位）

MSK 频带调制信号的载波的初始相位（单位：弧度）。

■ Output sample time (s)（输出信号的抽样周期）

MSK 频带调制器产生的调制信号的抽样周期（单位：秒）。

MSK 频带解调器（MSK Demodulator Passband）对 MSK 频带调制信号进行解调，得到二进制信息序列（或双极性信号序列）。图 8-89 所示是 MSK 频带解调器模块及其参数设置对话框。

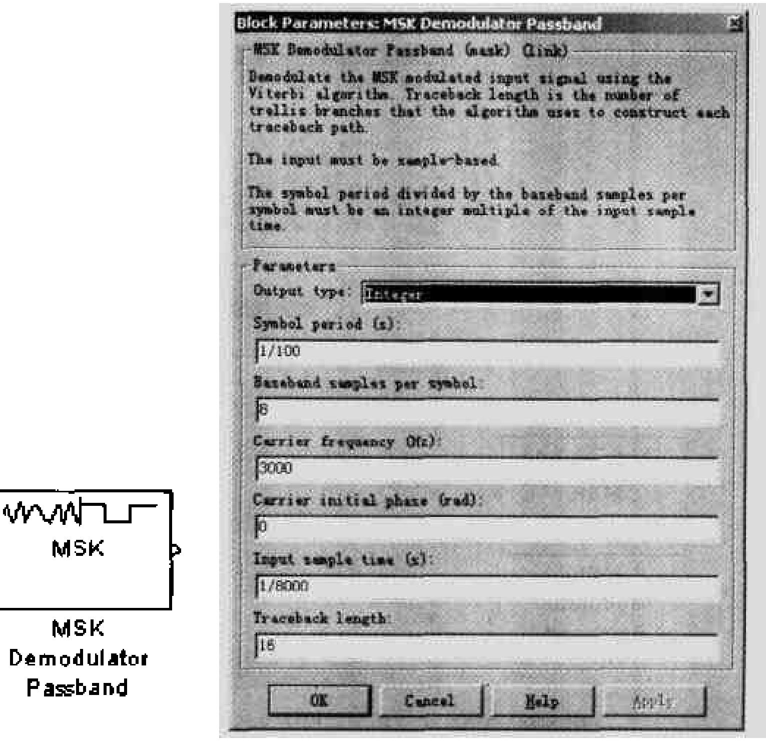



图 8-89 MSK 频带解调器模块及其参数设置对话框

在 MSK 频带解调器中，输入的频带调制信号先通过一个频率转化器变换成基带调制信号，然后通过一个 MSK 基带解调器把这个基带信号变换成基带解调信号。MSK 频带解调器模块是由 CPM 频带解调器模块构成的，它把 CPM 基带解调器模块的 M 相数（M-ary number）设置为 2，输出信号类型（Output type）设置为 Integer，调制指数（Modulation index）设置为 0.5，整形滤波器（Frequency pulse shape）设置为 Rectangular，脉冲长度（Pulse length）设置为 1，预输入信号（Symbol prehistory）等于 1。

MSK 频带解调器有以下几个参数。

■ Output type（输出信号类型）

MSK 频带解调器模块输出信号的类型。当本参数设置为 Integer 时，MSK 频带解调器模

块的输出信号是一个双极性信号（±1）；当本参数设置为 Bit 时，MSK 频带解调器模块的输出信号是二进制信号（0 或 1）。

■ Symbol period(s)（输出符号周期）

MSK 频带解调器模块输出信号的符号类型（单位：秒）。

■ Baseband samples per symbol（基带符号抽样数）

MSK 频带解调器模块中每个输入的频带调制信号在转化成基带调制信号后的抽样点的个数。

■ Carrier frequency (Hz)（载波频率）

MSK 频带调制信号的载波频率（单位：Hz）。

■ Carrier initial phase (rad)（载波相位）

MSK 频带调制信号的载波的初始相位（单位：弧度）。

■ Input sample time (s)（输出信号的抽样周期）

MSK 频带解调器的输入信号的抽样周期（单位：秒）。

■ Traceback length（回溯长度）

MSK 频带解调器模块中的 Viterbi 译码器的回溯长度。

## 8.7.5 基带 GMSK 调制

移动通信系统对信号带外辐射功率具有比较严格的要求，一般要求衰减幅度在 70dB 到 80dB 以上。虽然 MSK 调制信号的功率谱在主瓣以外衰减较快，但是仍然不能满足这种要求。高斯最小频移键控（GMSK，Gaussian Minimum Shift Keying）对 MSK 调制方式作出了改进，它通过在 MSK 调制器之前加入一个高斯低通滤波器来获得更加紧凑的频谱，因而在移动通信系统中获得了广泛的应用。

GMSK 调制过程中使用的高斯低通滤波器影响着 GMSK 调制信号的性能。高斯低通滤波器一般用参数 $B\cdot T$ 来表示，其中 $B$ 是高斯低通滤波器的归一化 3dB 带宽，$T$ 是码元长度。当 $B\cdot T=\infty$ 时，GMSK 调制信号就变成 MSK 调制信号。GMSK 调制信号的频谱随着 $B\cdot T$ 的减小而变得紧凑起来，但是这种频谱特性的改善是通过降低误比特率性能换来的。高斯低通滤波器的带宽越窄，输出的功率谱就越紧凑，误比特性能就变得越差。一般情况下，当 $B\cdot T$ 等于 0.25 时，GMSK 调制信号具有比较理想的性能。

GMSK 基带调制器（GMSK Modulator Baseband）对输入信号实施高斯最小频移键控调制，产生基带调制信号。GMSK 基带调制器模块及其参数设置对话框如图 8-90 所示。

GMSK 基带调制器模块实际上是一个 CPM 基带调制器模块，它把 CPM 基带调制器模块的 M 相数（M-ary number）设置为 2，输入信号类型（Input type）设置为 Integer，调制指数（Modulation index）设置为 0.5，整形滤波器（Frequency pulse shape）设置为 Gaussian。

GMSK 基带调制器模块主要有以下几个参数。

■ Input type（输入信号类型）

GMSK 基带调制器模块输入信号的类型。当本参数设置为 Integer 时，GMSK 基带调制器模块的输入信号是一个双极性信号（±1）；当本参数设置为 Bit 时，GMSK 基带调制器模

块的输入信号是二进制信号（0 或 1）。

■ BT product（BT 乘积）

GMSK 基带调制器中高斯低通滤波器的带宽和时间的乘积 $B \cdot T$ 。

■ Pulse length (symbol intervals)（脉冲长度）

GMSK 基带调制器中高斯低通滤波器的周期。当本参数等于 $k$ 时，高斯低通滤波器的周期等于输入符号周期的 $k$ 倍。

■ Symbol prehistory（预输入符号）

GMSK 基带调制器在仿真开始之前的输入符号。如果本参数是一个向量，则该向量的长度等于参数 Pulse length 的数值减 1。

■ Phase offset (rad)（相位偏移）

GMSK 基带调制信号的初始相位（单位：弧度）。

■ Samples per symbol（输出信号采样数）

对应于每一个输入符号（整数或二进制向量）GMSK 基带调制器产生的输出信号的抽样点的个数。

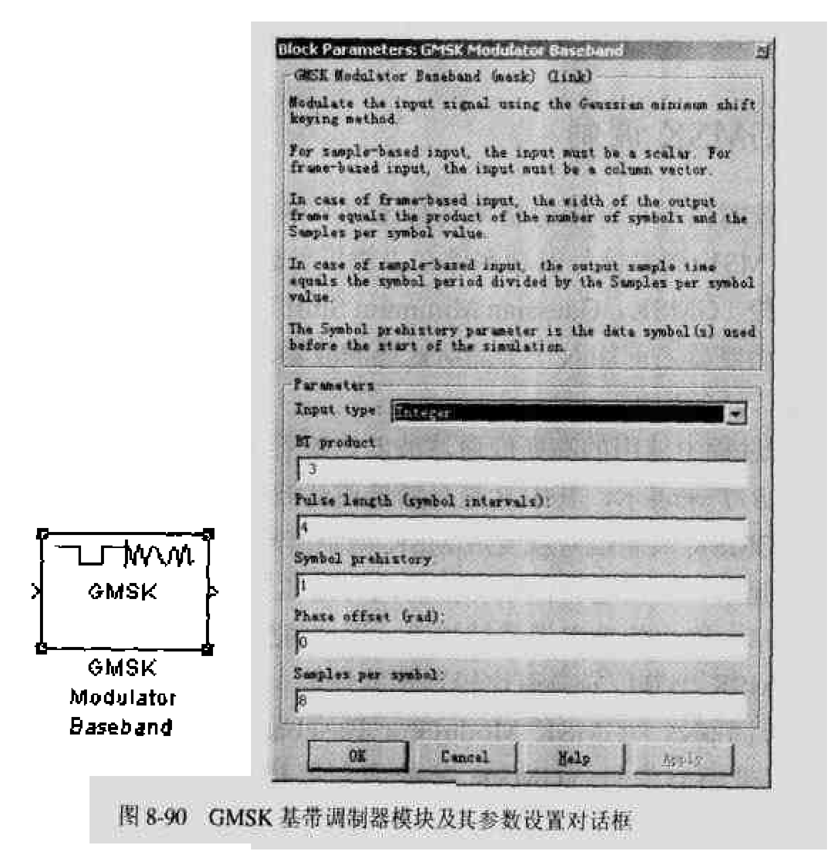



图 8-90　GMSK 基带调制器模块及其参数设置对话框

GMSK 基带解调器（GMSK Demodulator Baseband）对 GMSK 基带调制信号进行解调，得到原始的信息序列。GMSK 基带解调器模块及其参数设置对话框如图 8-91 所示。

GMSK 基带解调器模块实际上是一个 CPM 基带解调器模块，它把 CPM 基带解调器模块的 M 相数（M-ary number）设置为 2，输出信号类型（Output type）设置为 Integer，调制指数（Modulation index）设置为 0.5，整形滤波器（Frequency pulse shape）设置为 Gaussian。GMSK 基带解调器模块中的参数与 CPM 基带解调器模块的参数相同。

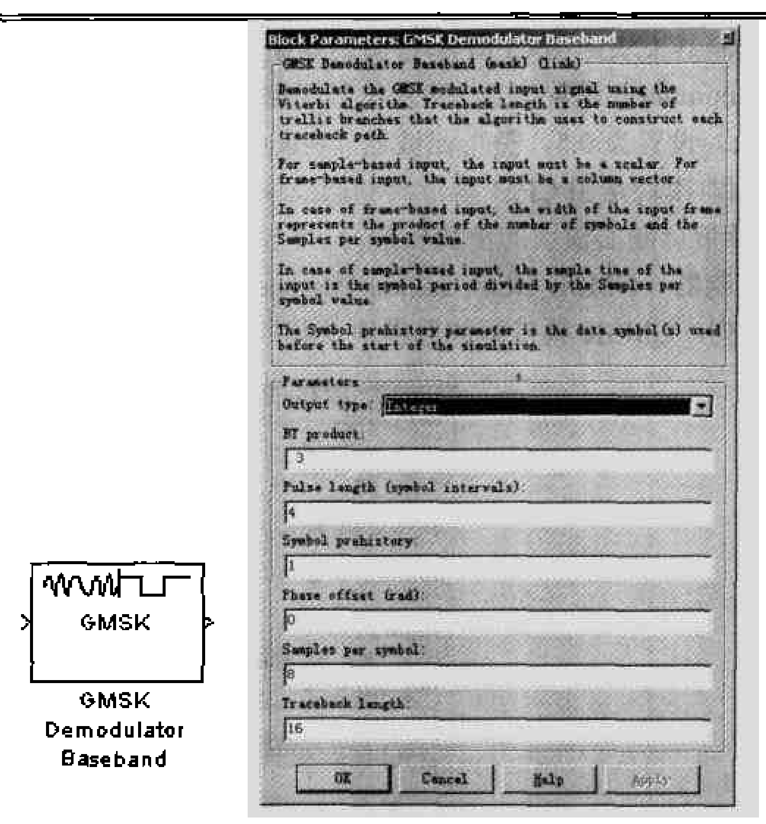



图 8-91　GMSK 基带解调器模块及其参数设置对话框

## 8.7.6　频带 GMSK 调制

GMSK 频带调制器模块对输入信号实施 GMSK 调制，产生频带调制信号。GMSK 频带调制器模块首先通过 GMSK 基带调制器把输入信号转换成基带调制信号，然后把这个基带信号调制到高频载波上，形成频带调制信号。GMSK 频带调制器模块及其参数设置对话框如图 8-92 所示。

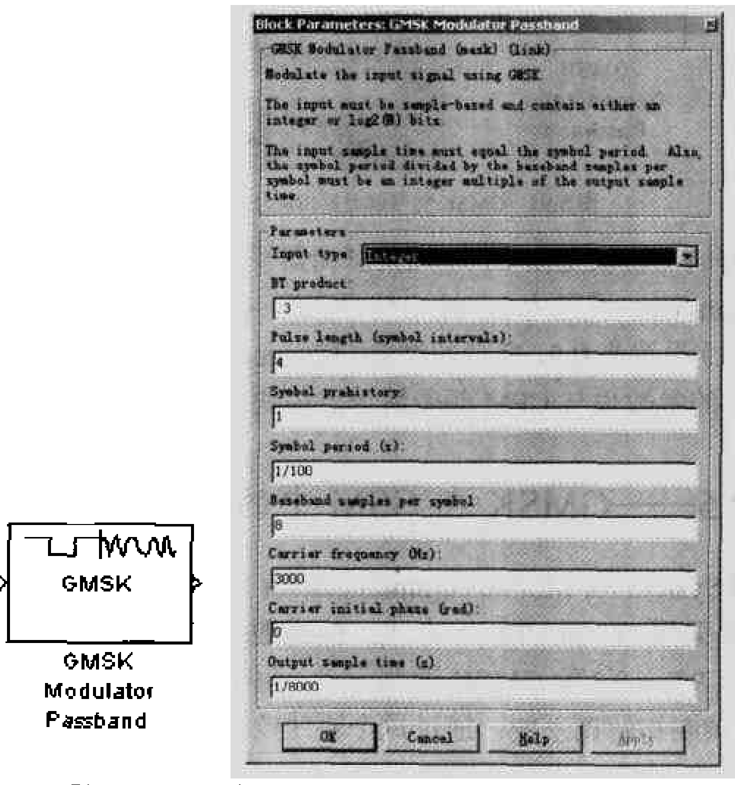



图 8-92　GMSK 频带调制器模块及其参数设置对话框

GMSK 频带调制器模块是由 CPM 频带调制器模块构造的，其中，CPM 频带调制器模块的 M 相数（M-ary number）设置为 2，输入信号类型（Input type）设置为 Integer，调制指数（Modulation index）设置为 0.5，整形滤波器（Frequency pulse shape）设置为 Gaussian。由于 GMSK 频带调制器模块是由 CPM 频带调制器模块构造的，因此 GMSK 频带调制器模块中的参数也与 CPM 频带调制器模块的参数相同，具体参数的含义请参照 CPM 频带调制器模块。

GMSK 频带调制器（GMSK Demodulator Passband）对 GMSK 频带调制信号进行解调，得到原始的信息序列。GMSK 频带解调器模块及其参数设置对话框如图 8-93 所示。

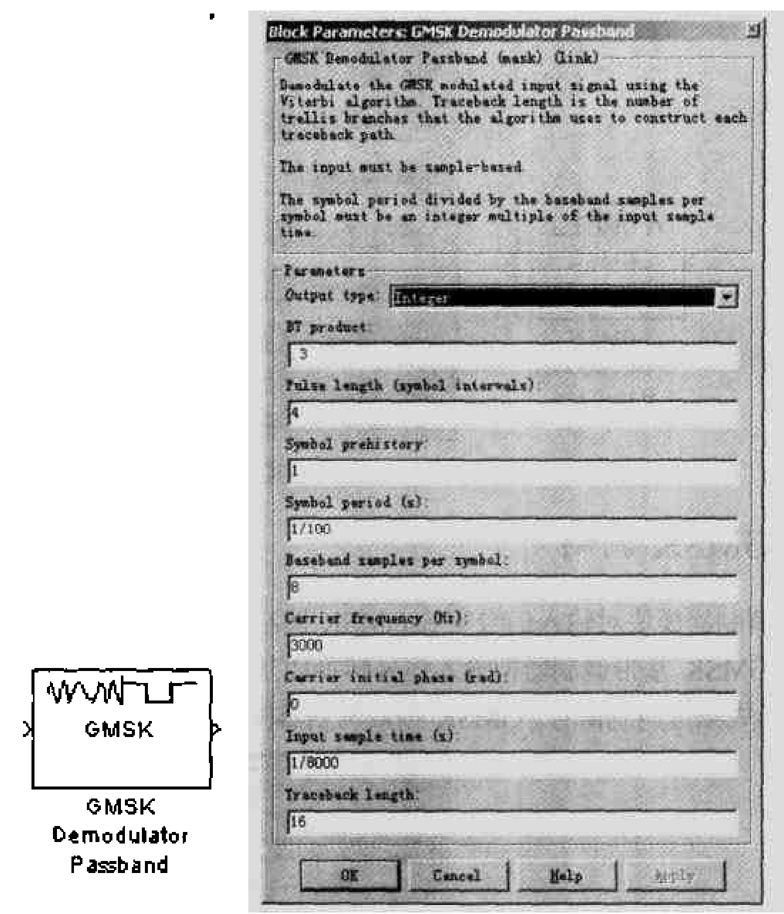



图 8-93　GMSK 频带解调器模块及其参数设置对话框

GMSK 频带解调器模块是由 CPM 频带解调器模块构造的，其中 CPM 频带解调器模块的 M 相数（M-ary number）设置为 2，输出信号类型（Output type）设置为 Integer，调制指数（Modulation index）设置为 0.5，整形滤波器（Frequency pulse shape）设置为 Gaussian。GMSK 频带解调器模块中的参数也与 CPM 频带解调器模块中的同名参数相同。

## 8.7.7　实例 8.5——GMSK 在 GSM 中的应用

GSM（Global System for Mobile communications）是目前世界上应用最为广泛的一种数字移动通信系统，它采用 TDMA/FDD 方式，信道数据传输速率达到 270.833 kbit/s（1625/6 kbit/s）。GSM 采用 $BT=0.3$ 的 GMSK 调制方式，每个信道的带宽等于 200 kHz。本节我们将设计一个 GMSK 调制和解调系统，通过这个系统绘制 GMSK 调制信号误码率与信噪比的关系曲线。图 8-94 所示是这个仿真模型的系统结构框图。

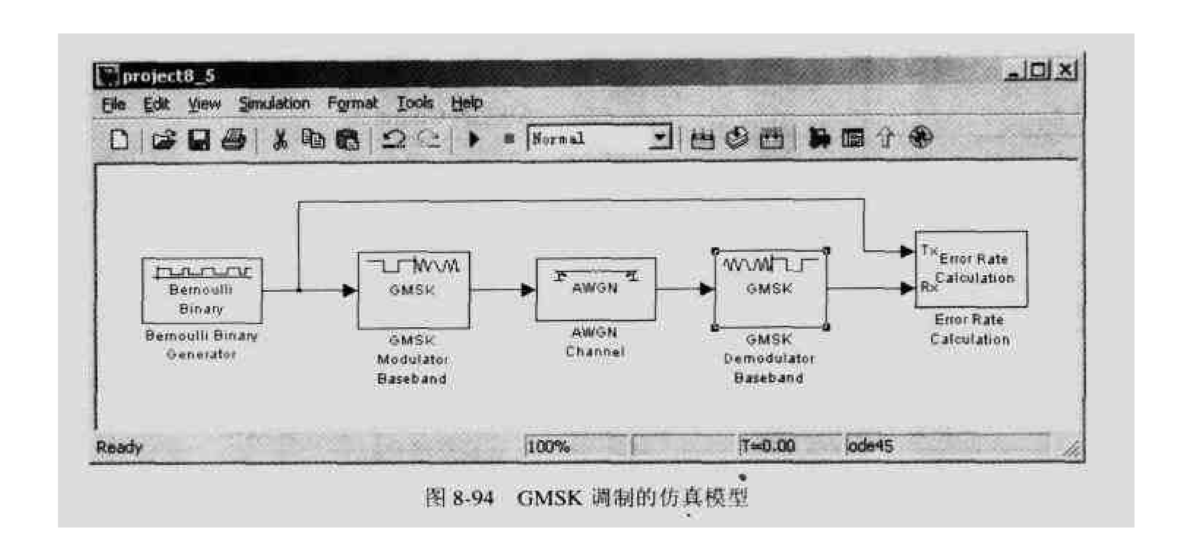



图 8-94　GMSK 调制的仿真模型

在图 8-94 所示的 GMSK 调制和解调系统中，Bernoulli Binary Generator（贝努利二进制序列产生器）产生一个二进制序列，然后通过 GMSK Modulator Baseband（GMSK 基带调制器模块）得到基带调制信号。表 8-27 和表 8-28 列出了这两个模块的参数设置情况。

**表 8-27　Bernoulli Binary Generator（贝努利二进制序列产生器）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Bernoulli Binary Generator |
| Probability of a zero | 0.5 |
| Initial seed | xInitialSeed |
| Sample time | xSampleTime |
| Frame-based outputs | Unchecked |
| Interlpret vector parameters as 1-D | Unchecked |

**表 8-28　GMSK Modulator Baseband（GMSK 基带调制器模块）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | GMSK Modulator Baseband |
| Input type | Bit |
| BT product | 0.3 |
| Pulse length (symbol intervals) | 4 |
| Symbol prehistory | 1 |
| Phase offset (rad) | 0 |
| Samples per symbol | 1 |

GMSK 基带调制信号首先通过一个 AWGN Channel（加性高斯白噪声模块），然后由 GMSK Demodulator Baseband（GMSK 基带解调器模块）对其实施解调，得到二进制解调信号，最后由 Error Rate Calculation（误码率统计模块）统计 GMSK 调制信号的误码率。表 8-29、表 8-30 和表 8-31 分别是加性高斯白噪声模块、GMSK 基带解调器模块以及误码率统计模块的参数设置。

表 8-29　　**AWGN Channel（加性高斯白噪声模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | AWGN Channel |
| Initial seed | 67 |
| Mode | Signal to noise ratio (SNR) |
| SNR (dB) | xSNR |
| Input signal power (watts) | 1 |

表 8-30　　**GMSK Demodulator Baseband（GMSK 基带解调器模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | GMSK Demodulator Baseband |
| Input type | Bit |
| BT product | 0.3 |
| Pulse length (symbol intervals) | 4 |
| Symbol prehistory | 1 |
| Phase offset (rad) | 0 |
| Samples per symbol | 1 |
| Traceback length | xTracebackLength |

表 8-31　　**Error Rate Calculation（误码率统计模块）的参数设置**

| 参数名称 | 参数值 |
| --- | --- |
| 模块类型 | Error Rate Calculation |
| Receive delay | xTracebackLength+1 |
| Computation delay | 0 |
| Computation mode | Entire frame |
| Output data | Workspace |
| Variable name | xErrorRate |
| Reset port | Unchecked |
| Stop simulation | Unchecked |

在 M 文件 project8_5main.m 中，我们对仿真模型用到的各个变量进行定义，然后通过 Simulink 重复运行仿真模型，从仿真结果中得到不同信噪比条件下 GMSK 调制信号的误码率。M 文件 project8_5main.m 的代码如下：

```
% 设置调制信号的抽样间隔
xSampleTime = 1/10000;
% 设置仿真时间的长度
xSimulationTime = 10;
% 设置随机数产生器的初始化种子
xInitialSeed = 61;
% 设置 GMSK 解调器的回溯长度
xTracebackLength = 4;
```

```
% x 表示信噪比的取值范围
x = 0:10;
% y 表示 GMSK 调制的误比特率
y = x;

for i = 1:length(x)
    % 信噪比依次取向量 x 的数值
    xSNR = x(i);
    % 执行 GMSK 仿真模型
    sim('project8_5');
    % 从 xErrorRate 中获得调制信号的误码率
    y(i) = xErrorRate(1);
end

% 绘制信噪比与误码率的关系曲线
semilogy(x,y);
```

在 MATLAB 工作区中输入命令行“project8_5”，启动仿真程序运行。图 8-95 所示是仿真结束之后得到的 GMSK 调制信号的误码率曲线。需要指出的一点是，GMSK 调制信号的误码率不仅取决于信号的信噪比，还跟 GMSK 调制器模块参数 Samples per symbol 以及 GMSK 解调器模块参数 Traceback length 有关。在我们这个仿真模型中，上述两个参数分别设置为 1 和 4。可以预见的是，如果增加这两个参数的数值，GMSK 调制信号的误码率将随之降低。在这种情况下，只有通过延长仿真时间或降低数据源的抽样速率才能够增加调制信号的个数，从而获得比较准确的误码率。同时，由于调制信号个数的增加，完成整个仿真过程所需的时间也随之增加。因此，仿真时间与仿真结果的精确度之间是相互矛盾的关系。

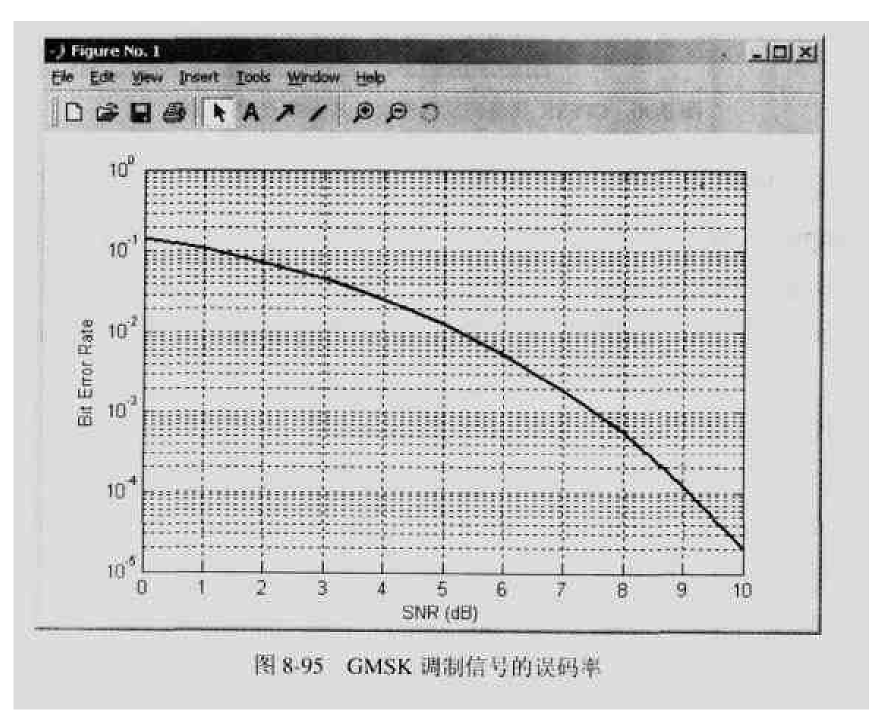



图 8-95　GMSK 调制信号的误码率

## 8.7.8 基带 CPFSK 调制

连续相位频移键控（CPFSK，Continuous Phase Frequency Shift Keying）是频移键控调制的一种改进形式，它产生连续相位的频率调制信号。CPFSK 基带调制器（CPFSK Modulator Baseband）对输入信号实施连续相位频移键控调制，产生基带调制信号。M 相 CPFSK 基带调制器的输入信号是介于-(*M*-1)和 *M*-1 之间的奇数，其中 $M=2^k$（$k=1,2,\cdots$）。CPFSK 基带调制器模块及其参数设置对话框如图 8-96 所示。

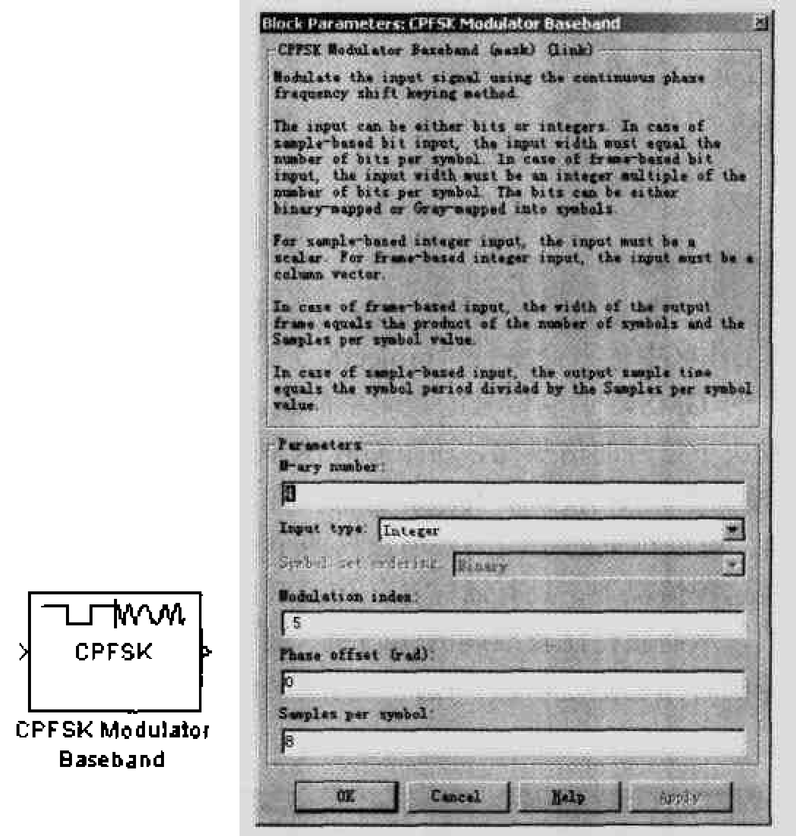



图 8-96　CPFSK 基带调制器模块及其参数设置对话框

CPFSK 基带调制器模块是由 CPM 基带调制器模块构成的。在 CPM 基带调制器模块中，输入信号类型（Input type）设置为 Integer，整形滤波器（Frequency pulse shape）设置为 Retangular，脉冲长度（Pulse length）设置为 1，预输入信号（Symbol prehistory）等于 1，相位偏移（Phase offset）等于 0。CPFSK 基带调制器模块的参数与 CPM 基带调制器模块中的相应参数相同。

CPFSK 基带解调器（CPM Demodulator Baseband）对 CPFSK 基带调制信号进行解调，它的输出信号是介于-(*M*-1)和 *M*-1 之间的奇数（或对应的长度为 k 的二进制向量），其中 *M* 是 CPFSK 基带调制器输入信号的相数，且 $M=2^k$（$k=1,2,\cdots$）。CPFSK 基带解调器模块及其参数设置对话框如图 8-97 所示。

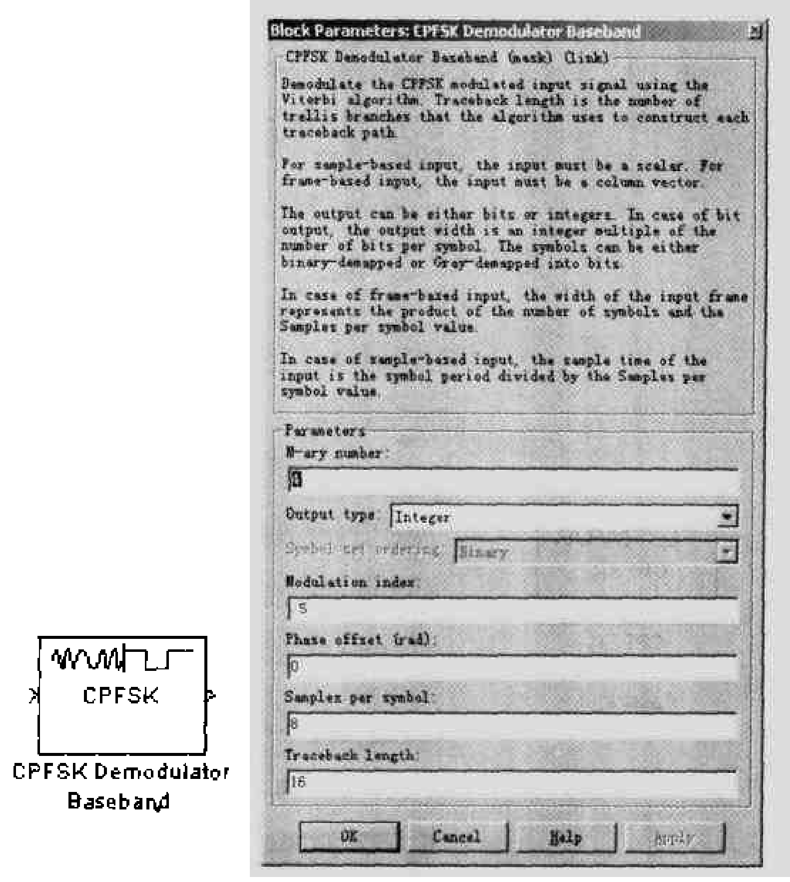



图 8-97　CPFSK 基带解调器模块及其参数设置对话框

CPFSK 基带解调器模块实际上是一个 CPM 基带解调器模块，它把 CPM 基带解调器模块的输出信号类型（Output type）设置为 Integer，整形滤波器（Frequency pulse shape）设置为 Retangular，脉冲长度（Pulse length）设置为 1，预输入信号（Symbol prehistory）等于 1。CPFSK 基带解调器模块的参数与 CPM 基带解调器模块中的相应参数相同。

## 8.7.9　频带 CPFSK 调制

CPFSK 频带调制器（CPFSK Modulator Passband）对输入信号实施连续相位频移键控调制，产生频带调制信号。当 CPFSK 频带调制器的输入信号的相数等于 $M=2^k$（$k=1,2,\cdots$）时，输入信号是介于-($M$-1)和 $M$-1 之间的奇数，这个输入信号既可以是整数，也可以是长度为 $k$ 的二进制向量。CPFSK 频带调制器模块及其参数设置对话框如图 8-98 所示。

CPFSK 频带调制器模块是由 CPM 频带调制器模块构成的，其中，CPM 频带调制器模块中的输入信号类型（Input type）设置为 Integer，整形滤波器（Frequency pulse shape）设置为 Retangular，脉冲长度（Pulse length）设置为 1，预输入信号（Symbol prehistory）等于 1。CPFSK 频带调制器模块的参数与 CPM 频带调制器模块中的相应参数相同。

CPFSK 频带解调器（CPM Demodulator Passband）对 CPFSK 频带调制信号进行解调，它的输出信号是介于-($M$-1)和 $M$-1 之间的奇数（或对应的长度为 $k$ 的二进制向量），其中 $M$ 是 CPFSK 频带调制器输入信号的相数，且 $M=2^k$（$k=1,2,\cdots$）。CPFSK 频带解调器模块及其参数设置对话框如图 8-99 所示。

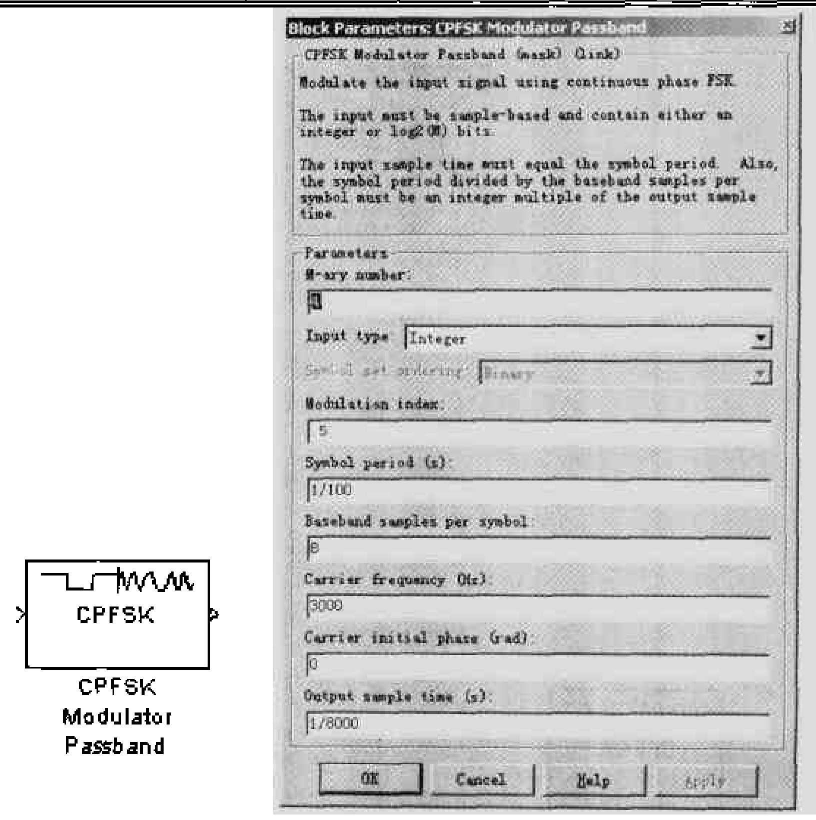



图 8-98　CPFSK 频带调制器模块及其参数设置对话框

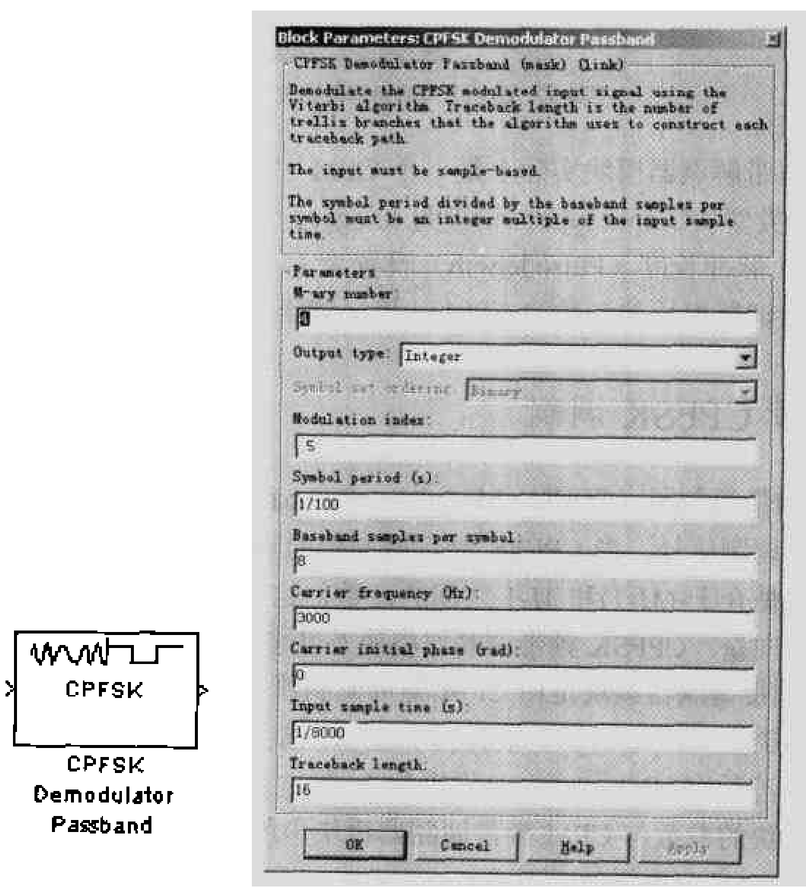



图 8-99　CPFSK 频带解调器模块及其参数设置对话框

在 MATLAB 中，CPFSK 频带解调器模块是由 CPM 频带解调器模块构造的，其中，CPM 频带解调器模块的输出信号类型（Output type）设置为 Integer，整形滤波器（Frequency pulse

shape）设置为 Retangular，脉冲长度（Pulse length）设置为 1，预输入信号（Symbol prehistory）等于 1。CPFSK 频带解调器模块的参数与 CPM 频带解调器模块中的相应参数相同。

# 第9章　仿真和调试

在完成仿真模型的设计工作之后，下一步是在计算机中运行这个仿真程序，从仿真过程中获得仿真数据，然后对这些仿真数据进行分析。然而，实际的仿真过程并不总是一帆风顺的，这个设计好的仿真模型可能会在运行的时候因为发生运行错误而被迫中断，或者仿真结束之后得到的数据与当初的设计目标相差甚远。因此，仿真后期的工作依然是比较繁重的，这些工作涉及到对仿真参数的设置，对仿真过程进行跟踪和调试以及如何采取最佳的方式对仿真数据进行处理等等问题。

## 9.1　运行仿真

一个仿真模块的运行实际上由两个步骤组成：设置仿真参数和运行仿真。由于仿真模块的不同特点，Simulink 的缺省参数设置并不一定适合于当前的仿真模型。仿真参数的设置在很大程度上影响着仿真的过程和结果，不恰当的参数设置甚至会得到与设计初衷完全相反的数据，因此，需要根据仿真模型的特征选择合适的参数，然后才能启动程序进行仿真。

### 9.1.1　设置仿真参数

在开始执行仿真之前，首先要设置相关的仿真参数，仿真参数在很大程度上影响着仿真的性能以及仿真结果的精度，不恰当的参数设定甚至会使仿真程序产生截然不同的结果。Simulink 的仿真参数通过仿真参数对话框进行设置，用户可以在仿真模块的菜单栏上选择 Simulation | Simulation parameters...打开仿真参数设置对话框。仿真参数设置对话框由 Solver、Workspace I/O、Diagnostics、Advanced 以及 Real-Time Workshop 5 个面板组成，在本节里将依次介绍这5个面板中各个参数的含义及其应用。

#### 1. Solver 属性页

通过 Solver 属性页可以设置一些基本的仿真参数，如设置仿真的开始时间和结束时间，确定求解的类型和精度，以及信号的输出方式。图 9-1 所示是 Solver 属性页。

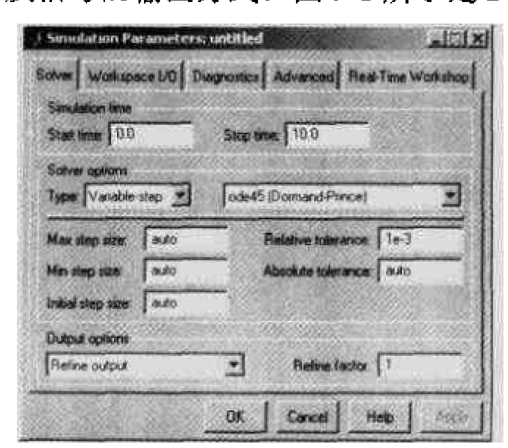



图 9-1　Solver 属性页

以下是图 9-1 所示中各参数的含义。

■ Simulation Time（仿真时间）

在 Solver 属性页中，用户可以设置仿真的开始时间（Start Time）和结束时间（Stop Time）。缺省情况下，仿真的开始时间设置为 0，结束时间等于 10s，即仿真时间持续 10s。当仿真的结束时间设置为 inf 时，仿真过程将无限期地延续下去，直到发生某种指定事件或用户终止这个过程。

需要注意的是，仿真时间并不是实际的时钟。例如，当把仿真时间设置为 10s 时，执行整个仿真过程需要的时间可能大于 10s，也可能小于 10s，这取决于仿真模型的复杂度、求解器的步长以及计算机的运行速度等因素。

■ Solvers（求解器）

Simulink 仿真模型实际上是对模型的输入信号、输出信号及其内部状态的一种计算过程，对此 MATLAB 提供了多种求解器，以满足不同模型的要求。这些求解器只适合于求解某些类型的仿真模型，对于同一种模型，不同的求解器可能需要不同的仿真时间，并且产生不同精度的数据。Simulink 提供了 4 种类型的求解器，即固定步长连续求解器（Fixed-step Continuous Solvers）、可变步长连续求解器（Variable-step Continuous Solvers）、固定步长离散求解器（Fixed-step Discrete Solver）以及可变步长离散求解器（Variable-step Discrete Solver）。

对于固定步长连续求解器（Fixed-step Continuous Solvers），Simulink 按照固定的步长计算仿真模型中的各个连续状态，并且根据状态的导数采用数字积分方式计算连续状态的数值。在 MATLAB 中，固定步长连续求解器可选的算法包括 ode5（Dormand-Prince 算法）、ode4（4 阶 Runge-Kutta 算法）、ode3（Bogacki-Shampine 算法）、ode2（Heun 算法，即改进型 Euler 算法）以及 ode1（Euler 算法）。

固定步长离散求解器（Fixed-step Discrete Solver）采用固定的步长进行仿真，但是它不采用积分方式，因此适合于没有连续状态的仿真模型。

对于可变步长连续求解器（Variable-step continuous solvers），如果仿真模型中连续状态的变化速度加快，Simulink 将自动降低仿真的步长以提高仿真的精度；反之，如果仿真模型中连续状态的变化速度减慢，Simulink 将加大仿真的步长，从而能够节省仿真过程所需的时间。可变步长连续求解器有以下几种算法：

ode45 算法：ode45 算法（即 Runge-Kutta(4,5)算法）是一种步长为 1 的求解器算法，在计算 $t_n$ 时刻的数值 $y(t_n)$ 时，Simulink 只需要 $t_{n-1}$ 时刻的数值 $y(t_{n-1})$。对于大多数仿真模型来说，ode45 算法是最优的一种算法，也是仿真模型的缺省设置。

ode23 算法：ord23 算法（即 Runge-Kutta(2,3)算法）也是一种步长为 1 的求解器算法，它在具有较低的精度要求和较小的信号变化速率时能够得到比 ord45 算法更佳的效果。

ode113 算法：ord113 是一种可变阶 Adams-Bashforth-Moulton PECE 求解器算法，在计算 $t_n$ 时刻的数值 $y(t_n)$ 时，Simulink 需要多个该时刻之前的数值 $y(t_{n-1}), y(t_{n-2})\cdots$。ord113 算法在信号变化速率较高时优于 ord45 算法。

ode15s 算法：ode15s 算法是一种基于数字差分公式（NDF，Numerical Differentiation

Formulas）的可变阶求解器算法。如果仿真模型具有很高的信号变化速率，或者 ord45 算法不能取得令人满意的效果，这时候可以尝试 ode15s 算法。当选择 ode15s 算法时，用户可以在 Maximum order 下拉列表中选择本算法的最大阶数（介于 1 和 5 之间的整数）。高阶算法的优点是准确性较高，但是显得不够稳定，因此要根据实际需要选择合适的阶数。

ode23s 算法：ode23s 算法是一种基于二阶修正 Rosenbrock 公式的求解器算法。由于 ode23s 算法的步长为 1，它在精度较低的仿真过程中具有比 ode15s 算法更高的效率。

ode23t 算法：ode23t 是一种梯形内插算法，它适用于信号变化速率不高的场合。

ode23tb 算法：ode23tb 是一种 TR-BDF2 算法，它的前端采用梯形方式的 Runge-Kutta 公式，后端采用二阶后向差分公式（BDFs，Backward Differentiation Formulas）。与 ode23s 算法类似地，ode23tb 算法在精度较低的仿真过程中具有比 ode15s 算法更高的效率。

可变步长离散求解器（Variable-step discrete solver）与可变步长连续求解器类似，它能够自动调整仿真步长，但是不进行积分，因此适合于没有连续状态的仿真模型。当仿真模型中没有连续状态时，如果用户选择可变步长求解器，则缺省情况下将采用可变步长离散求解器。

总的说来，可变步长求解器比固定步长求解器具有更高的仿真效率。在固定步长求解器中，即使仿真模型没有触发任何事件，Simulink 也需要执行依次计算；而在可变步长求解器中，Simulink 将根据实际情况决定是执行计算操作。

作为可变步长求解器比固定步长求解器仿真效率的一个比较，我们来看看一个仿真模型。图 9-2 所示是这个仿真模型的示意图。在这个模型中我们设置了两个正弦信号产生器（Sine Wave 和 Sine Wave1），其中 Sine Wave 的抽样间隔是 0.5 秒，Sine Wave1 的抽样间隔是 0.75 秒，这两个信号分别输入到一个单位时延模块（Unit Delay 和 Unit Delay1）。

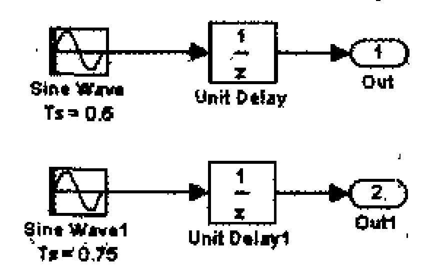



图 9-2　可变步长求解器与固定步长求解器的比较

对于图 9-2 所示的仿真模型，由于两个正弦信号具有不同的抽样间隔（0.5 秒和 0.75 秒），如果采用固定步长求解器，Simulink 需要每隔 0.25 秒计算一次输出信号，这时候 Simulink 需要计算的时刻是[0,0.25,0.5,0.75,1,1.25,1.5,1.75,2⋯]。而对于可变步长求解器，由于仿真模型在 $t = 0.25$ 时刻没有发生任何时间，因此 Simulink 只在[0,0.5,0.75,1,1.5,2,2.25⋯]等时刻进行计算，从而减轻了仿真过程的计算量，提高了仿真效率。

■　Step Sizes（仿真步长）

对于可变步长求解器，用户可以设置最大仿真步长、初始仿真步长以及最小仿真步长，这三个参数的缺省值都是 auto。对于固定步长求解器，用户可以设置固定仿真步长参数，它的缺省值也是 auto。

最大仿真步长（Maximum step size）用于设置可变步长求解器的最大求解间隔。缺省情况下，MATLAB 通过公式 $h_{max} = (t_{stop} - t_{start})/50$ 来计算最大仿真步长，其中 $t_{stop}$ 和 $t_{start}$ 分别表示仿真的开始时间和结束时间。这个数值对于一般的仿真过程都显得足够大。如果仿真的

时间比较长，则可以按照需要把这个数值设置的小一些。当仿真模型执行周期性操作时，可以把这个参数设置为这个周期的若干分之一。

初始仿真步长（Initial step size）是用户给 Simulink 仿真步长的一个“建议值”。如果这个初始仿真步长过大，使得仿真结果不能达到所需的误差标准，Simulink 将降低初始仿真步长的数值。

最小仿真步长（Minimum step size）是 Simulink 仿真步长的最小值。当 Simulink 需要降低仿真步长以满足仿真结果的误差要求，且调整后的仿真步长低于这个最小仿真步长时，Simulink 将发出一个警告。本参数可以设置为一个实数，也可以是一个长度为 2 的向量，其中的一个元素表示最小仿真步长，第二个元素表示在 Simulink 发出错误通知之前可以容许的最小仿真步长警告数目。如果把向量的第二个元素设置为 0，Simulink 在第一次发生最小仿真步长警告时就认为发生了错误。如果这个元素设置为-1，则 Simulink 将把所有的警告都不当作错误。缺省情况下最小仿真步长取决于计算机的精度，同时 Simulink 忽略所有的最小仿真步长警告。

对于固定步长求解器，Simulink 以固定的间隔执行仿真计算，这个间隔由固定仿真步长参数确定。在固定步长仿真过程中，Simulink 设置了 3 种仿真模式：多任务方式（MultiTasking）、单任务方式（SingleTasking）以及自动方式（Auto）。多任务方式适用于实时多任务系统，Simulink 在检测到不同模块之间抽样速率不匹配时认为是发生了错误。因此，在多任务方式中，不同抽样速率之间应该用速率转换模块（Rate Transition）进行连接。单任务方式适用于单任务系统，这时候 Simulink 不需要检查模块之间的速率匹配问题。自动方式则根据仿真模块的内部结构选择单任务或多任务方式：如果所有模块具有相同的速率，Simulink 采用单任务方式；否则，Simulink 采用多任务方式。

■　Error Tolerances（误差容限）

Simulink 的误差容限由两个参数确定，即相对误差 *rtol*（relative tolerance）和绝对误差 *atol*（absolute tolerance）。在每一步仿真过程中，Simulink 要求每一个状态 $i$ 的当前误差 $e_i$ 满足条件 $e_i \le \max(atol, rtol \times |x_i|)$，其中 $x_i$ 是状态 $i$ 的当前值。图 9-3 所示是关于相对误差范围和绝对误差范围的示意图。

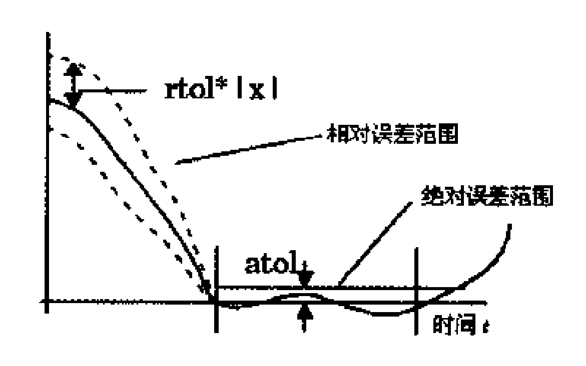



图 9-3　相对误差和绝对误差

在仿真过程中 Simulink 求解器计算每个状态的当前值及其误差 $e_i$。如果某个状态的误差不满足这个误差容限，Simulink 在降低仿真步长之后重新计算一次。缺省情况下相对误差等于 $10^{-3}$，绝对误差等于 auto（这时候 Simulink 把绝对误差设置为 $10^{-6}$）。在仿真过程中，Simulink 可以根据仿真结果重新设置这个绝对误差，使之等于到当前时刻为止 $atol \times |x_i|$ 的最大值。

■ Output Options（输出信号选项）

输出信号选项允许用户指定 Simulink 输出信号的方式：提取输出信号（Refine output）、产生额外输出信号（Produce additional output）以及产生指定的输出信号（Produce specified output only）。

提取输出信号选项（Refine output）使得 Simulink 不需要改变仿真步长而能够增加仿真结果精度。当本参数设置为 $k$ 时，Simulink 在相邻两个仿真时刻之间插入 $k-1$ 个输出信号，从而产生 $k$ 倍的输出信号数。缺省情况下本参数设置为 1。提取输出信号选项适用于可变步长求解器，尤其是 ode45 算法。如果采用 ode45 算法时得到的输出结果不够平滑，这时候可以考虑增加 Refine output 的数值。

产生额外输出信号选项（Produce additional output）使得 Simulink 在按照指定的求解器产生输出信号时产生额外抽样时刻的输出信号；产生指定的输出信号选项（Produce specified output only）则要求 Simulink 只产生指定时刻的输出信号。这两种方式都将改变 Simulink 的仿真步长。例如，对于一个在[0,2.5,5,8.5,10]时刻产生输出信号的仿真模型，如果把 Refine output 设置为 2，则 Simulink 在[0,1.25,2.5,3.75,5,6.75,8.5,9.25,10]时刻产生输出信号；如果把 Produce additional output 参数设置为[0:10]，则 Simulink 在[0,1,2,3,4,5,6,7,8,9,10]时刻产生输出信号，同时不同的求解器可能会产生其他的时刻产生输出；如果把 Produce Specified Output Only 参数设置为[0:10]，则 Simulink 只在[0,1,2,3,4,5,6,7,8,9,10]时刻产生输出信号。一般情况下，指定的输出时刻应该是基本仿真步长的整数倍。

## 2. Workspace I/O 属性页

Workspace I/O 属性页用于设置仿真的输入信号和输出信号选项，如图 9-4 所示。在 MATLAB 中，Simulink 可以从工作区中读取输入信号和初始状态，也可以把输出信号和状态保存到工作区中。通过这种方式，Simulink 可以对特定的输入数据进行仿真，也可以从上次仿真的中断点恢复运行。

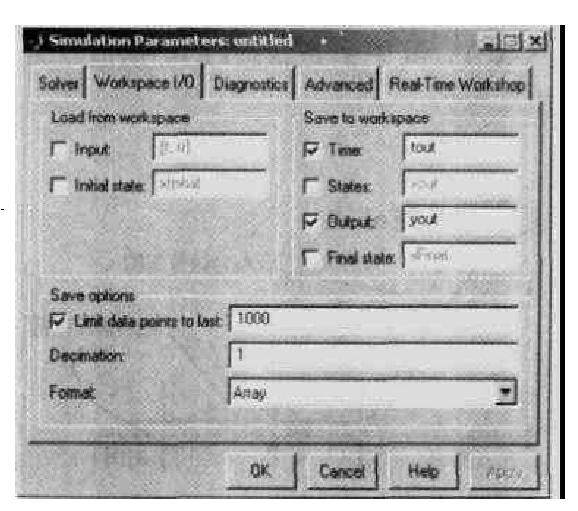



图 9-4 Workspace I/O 属性页

在 Workspace I/O 属性页中，选中与 Input 参数对应的复选框，然后在旁边的输入框中填写与输入信号相对应的工作区变量的名称，则 Simulink 在仿真过程中将从工作区读取输入数据。缺省的输入信号是一个行向量$[t,u]$，其中 $t$ 是表示抽样时间点的列向量，$u$ 则是输入信号的数值，其中每一行的元素是与 $t$ 相对应的时刻的抽样值。

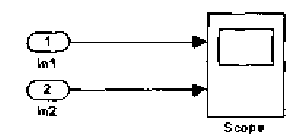



图 9-5　一个简单的仿真模型

例如，对于图 9-5 所示的仿真模型，它需要从工作区获取输入数据。如果把 Input 参数设置为$[t,u]$，且$t=[1:0.01:10]'$，$u=[\sin(t),\cos(t)]$，则第一个输入端口（In1）的输入信号是一个正弦信号，第二个输入端口（In2）的输入信号则是一个余弦信号。图 9-6 所示是当仿真时间分别设置为 10 秒和 20 秒时示波器的输出信号。从图 9-6 左图所示中可以看到，在仿真时刻 0，两个输出信号并不等于$\sin 0$或$\cos 0$，这是因为时间向量$t$的起始时刻是 1 秒。对于这种情况，Simulink 采用外推方式求得时刻 0 的输出信号。同样地，当仿真的结束时间设置为 20 秒时，由于时间向量$t$的结束时刻是 10 秒，Simulink 也将采用外推方法来求解，因此图 9-6 右图所示中 10 秒以后的输出信号是一条直线。

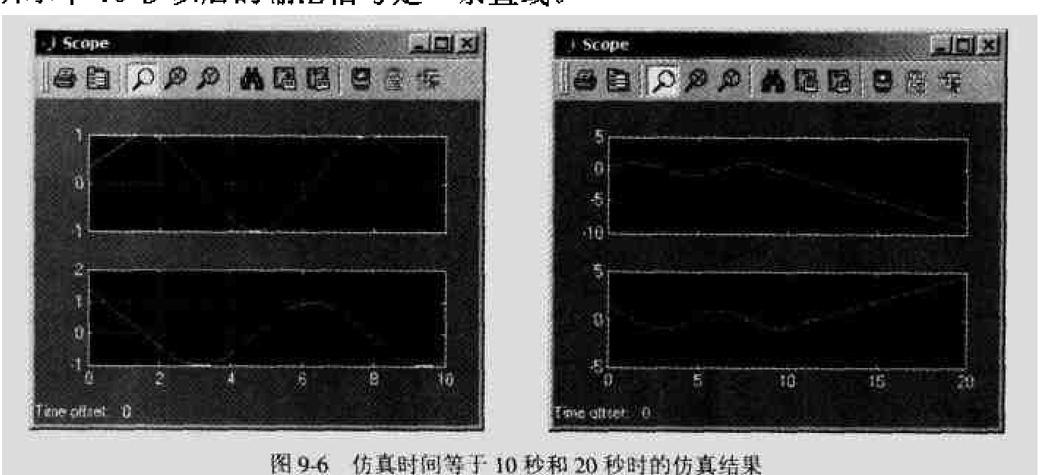



图 9-6　仿真时间等于 10 秒和 20 秒时的仿真结果

另外，Input 参数也可以设置为一个带时间的结构（Structure with Time），这种结构中有两个子结构 time 和 signals。例如，图 9-5 所示的仿真模型也可以用带时间的结构来表示。将 Input 参数设置为$s$，并且在工作区中输入如下代码：

```
>> t=[1:0.01:100];
>> s.time=t;
>> s.signals(1).values=sin(t);
>> s.signals(1).dimensions=1;
>> s.signals(2).values=cos(t);
>> s.signals(2).dimensions=1;
```

Input 参数也可以用一系列的工作区变量来表示，这些变量之间用逗号分割。例如，对于上面的示例，可以把 Input 参数设置为$s1,s2$，在工作区中输入如下代码，然后运行仿真程序，这时候将得到相同的仿真结果。

```
>> t=[1:0.01:10]';
>> s1.time=t;
>> s1.signals.values=sin(t);
>> s1.signals.dimensions=1;
```

```
>> s2.time=t;
>> s2.signals.values=cos(t);
>> s2.signals.dimensions=1;
```

仿真模块的初始状态也可以从工作区中输入。在 Workspace I/O 属性页中选中与 Initial state 参数对应的复选框，然后在旁边的输入框中填写与保存了初始状态的工作区变量的名称，则 Simulink 在仿真开始的时候首先从工作区读取数据作为仿真模块的初始状态。初始状态变量的输入方式与输入信号类似，即可以是一个向量，也可以是一个结构，它的缺省值是 *xInitial* 。

仿真结束时，Simulink 可以将仿真结果保存到工作区中，用于对仿真结果进行分析，或者保存起来作为下一次仿真的输入信号。在 Workspace I/O 属性页中，可以保存的参数包括时间（Time，缺省变量名称为 *tout* ）、状态（States，缺省变量名称为 *xout* ）、输出信号（Outputs，缺省变量名称为 *yout* ）以及结束状态（Final state，缺省变量名称为 *xFinal* ）。如果选中与这些参数对应的复选框，并且把它们设置为相应的变量名称，则 Simulink 将根据设定的格式把输出信号保存到工作区中。

通过 Workspace I/O 属性页中的 Save options 选项，用户可以设置数据保存的方式。当选中了与 Limit data points to last 相对应的复选框，并且将该参数设置为 $N$ 时，如果保存的数据超过 $N$，Simulink 将忽略最初的若干输出信号，只保存最后 $N$ 个数据。Decimation 参数指定了输出向量的抽取方式。如果 Decimation 参数等于 1，则 Simulink 将每个输出信号都保存到工作区变量中；当 Decimation 参数等于 $k$ 时，Simulink 每隔 $k$ 个输出信号保存一个数据。输出信号的保存格式有 3 种：向量格式（Array）、带时间的结构（Structure with time）以及不带时间的结构（Structure）。在向工作区保存输出数据时，这些参数适用于所有工作区变量，用户可以根据自己的需要设置相应的保存方式。

### 3. Diagnostics 属性页

仿真过程中经常会遇到各种事件，如警告、错误等，用户可以通过 Diagnostics 属性页设置这些事件的处理方式。图 9-7 所示是 Diagnostics 属性页。

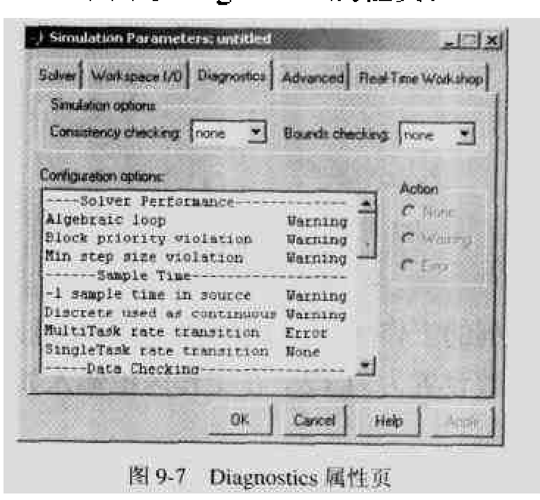



图 9-7 Diagnostics 属性页

以下是图 9-7 所示中各参数的含义。

■ Consistency Checking（一致性检查）

一致性检查主要对 S-函数的一致性进行检查，由于它在很大程度上降低了仿真的效率，

因此通常把本参数设置为 none。如果需要对仿真模块实施一致性检查，将本参数设置为 error 或 warning，则 Simulink 在检测到数据不一致时将触发错误或警告。在仿真过程中，为了提高仿真的效率，求解器一般将上一次的仿真结果保存起来供下一步仿真使用。如果在这个过程中数据发生了改变，Simulink 仍然将使用缓存中的数据进行计算，从而得到错误的结果。为了消除这种不一致性，需要对模块实施一致性检查，这时候 Simulink 在仿真的每一个步骤都将读取或重新计算信号的当前值，并且把这个数值与缓冲中的数值进行比较。如果比较的结果不相等，则 Simulink 将发出关于一致性检查的错误或警告。一般情况下一致性检查参数都设置为 off。

■ Bounds Checking（边界检查）

边界检查主要用于检查用户编写的 S-函数的内存写入错误。在用户自己编写 S-函数的过程中，一种常见的错误是对不属于程序的内存空间实施写操作，这种操作将破坏系统的稳定性，从而引发一个程序错误。为了检测这种错误，可以将本参数设置为 error 或 warning，这时候 Simulink 将对每一个模块实施边界检查，从而在很大程度上降低了仿真效率。如果不需要实施边界检查，应该本参数设置为 off。

■ Configuration options（配置选项）

通过配置选项用户可以对各种可能发生的事件的处理方式指定相应的反应措施。Simulink 可能采取的措施 Warning（警告）、Error（错误）或 None（忽略）。要改变事件的处理方式，只需在配置选项列表中选中相应的参数，然后在右边的 Action 栏中选取 Warning、Error 或 None。表 9-1 列出了配置选项的各种事件及其含义。

表 9-1　配置选项的事件及其定义

| 事件名称 | 事件定义 | 缺省设置 |
| --- | --- | --- |
| -1 sample time in source | 信号源的抽样时间等于-1 | Warning |
| Algebraic loop | 模块在仿真过程中出现代数环路 | Warning |
| Block Priority Violation | Simulink 在仿真过程中检测到模块包含优先级错误 | Warning |
| Check for singular matrix | Simulink 在使用 Product 模块执行矩阵除法运算时检测到奇异矩阵 | None |
| Data overflow | 信号的数值超过了所选数据类型的表达范围 | Warning |
| Discrete used as continuous | 单位延时模块 Unit Delay 从连续时间模块中继承了抽样时间 | Warning |
| int32 to float conversion | 32 位整数转化成浮点数可能会造成精度下降 | Warning |
| Invalid FcnCall Connection | Simulation 检测到错误的函数调用子模块 | Error |
| Min step size violation | 下一个仿真计算中使用的仿真步长低于模块中设置的最小仿真步长 | Warning |
| Multitask rate transition | Simulink 在多任务方式下检测到无效的速率转换 | Error |
| Parameter downcast | Simulink 在计算模块的输出信号时需要把数值转化为表示范围更小的数据类型，如把 32 位整数转换成 8 位整数 | Error |
| Parameter overflow | 模块中参数的数值超过了该参数的数据类型所能表达的范围 | Error |

续表

| 事件名称 | 事件定义 | 缺省设置 |
| --- | --- | --- |
| Parameter precision loss | Simulink 在计算模块的输出信号时需要把参数的数据类型转化成表示范围更小的数据类型，如把浮点数转换成 8 位整数 | Warning |
| S-function upgrades needed | 仿真模块中包含了低于 MATLAB 的当前版本的模块 | None |
| Signal label mismatch | Simulink 在仿真过程中检测到具有相同信号源和不同标签的虚拟信号 | None |
| SingleTask rate transition | 单任务模式下 Simulink 检测到两个模块之间需要进行速率转换 | None |
| Unconnected block input | 仿真模块中包含没有连接的输入端口 | Warning |
| Unconnected block output | 仿真模块中包含没有连接的输出端口 | Warning |
| Unconnected line | 仿真模块中包含没有连接的线条 | Warning |
| Underspecified data types | Simulink 在仿真过程中不能推断出信号的数据类型 | None |
| Unneeded type conversions | Simulink 在仿真过程中检测到不需要的数据类型转换模块 Data Type Conversion | None |
| Vector/Matrix conversion | 仿真模块的输入信号发生向量和矩阵之间的类型转换 | None |

4. Advanced 属性页

Advanced 属性页提供了其他对仿真过程有影响的高级参数设置，其中包括内置参数、性能优化参数、模块验证以及乘法器的硬件特征。Advanced 属性页如图 9-8 所示。

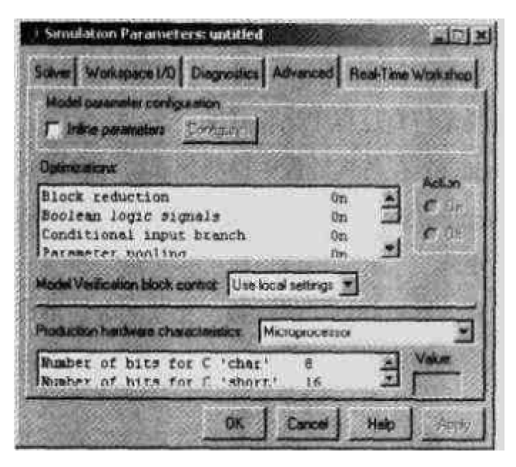



图 9-8 Advanced 属性页

以下是图 9-8 所示中各参数的含义。

■ Inline parameters（内置参数）

缺省情况下用户可以在仿真过程中改变各个模块的参数设置情况，但是在选中与内置参数（Inline Parameters）相对应的复选框之后，所有模块的参数都不能在仿真进行的过程中动态改变。使用内置参数的好处是 Simulink 可以把输出信号只取决于参数设置的模块从仿真循环中隔离开来，从而能够加快仿真的进度。另外，在很多情况下仿真模块的参数被设置为工作区中的变量，如果需要在仿真过程中动态改变这个参数的数值，可以把这个工作区变量声明为全局变量（Global）。为此，选中与内置参数相对应的复选框之后，单击旁边的 Configure

按钮，出现如图 9-9 所示的对话框。在这个对话框中，单击 Global (tunable) parameters 列表下面的 New 按钮可以添加一个全局变量。如果需要更改模块的参数值，只需在工作区中修改与这个参数相对应的工作区变量的数值，然后从菜单中选择 Edit | Update Diagram (Ctrl+D)，则模块参数更新为相应的数值。

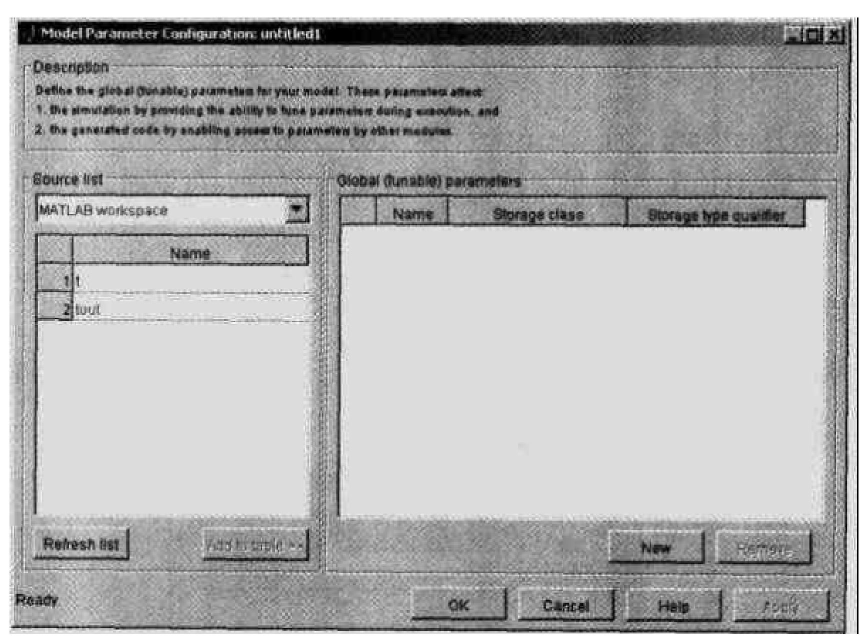



图 9-9 内置参数设置对话框

■ Optimizations（性能优化参数）

在仿真过程中，Simulink 还采用了一些措施来提高仿真的性能。表 9-2 列出了这些性能优化措施及其含义。

表 9-2 性能优化参数及其定义

| 参数名称 | 参数定义 | 缺省设置 |
|---|---|---|
| Block reduction | Simulink 把一组模块合并成一个模块，从而能够提高仿真速度 | On |
| Boolean logic signals | 当本参数设置为 on 时，如果仿真模块只能接收布尔型输入信号，则 Simulink 在检测到非布尔型输入信号时报错；否则，仿真模块可以接收其他类型的输入信号 | On |
| Conditional input branch | 本参数用于对包含 Switch 和 Multiport Switch 模块的仿真程序进行优化。当本参数设置为 on 时，Simulink 只计算这两个模块的输入数据信号和控制信号 | On |
| Parameter pooling | 本参数用于实时系统的代码生成过程 | On |
| Signal storage reuse | 当本参数设置为 off 时，Simulink 为不同模块的输出信号分配不同的缓冲区，从而需要占用较大的内存空间。本参数一般只在对 C-MEX S-函数进行调试或模块中包含浮动示波器或浮动显示器（floating Scope/Display）时设置为 off | On |
| Zero-crossing detection | 当本参数设置为 on 时，Simulink 在可变步长仿真过程中采用过零点检测方法动态调整仿真步长 | On |

■ Model Verification block control（模块验证）

Simulink 根据 Model Verification block control 的参数设置来决定是否对仿真模块实施验证，可供选择的方法有 Use local settings、Enable all 以及 Disable all。当选择 Use local settings 时，Simulink 根据各个模块的 Enable Assertion 参数来决定是否对这个模块进行验证；当选择 Enable all 时，Simulink 对所有的模块进行验证；当选择 Disable all 时，Simulink 对所有的模块都不实施验证。

■ Production Hardware Characteristics（乘法器的硬件特征）

如果需要通过 MATLAB 产生适合于实际数字系统的代码，可以通过 Production Hardware Characteristics 参数指定仿真采用的乘法器的类型及其位数。

#### 5. Real-Time Workshop 属性页

Real-Time Workshop 属性页主要用于设置一些关于实时仿真的参数，其参数设置界面如图 9-10 所示。

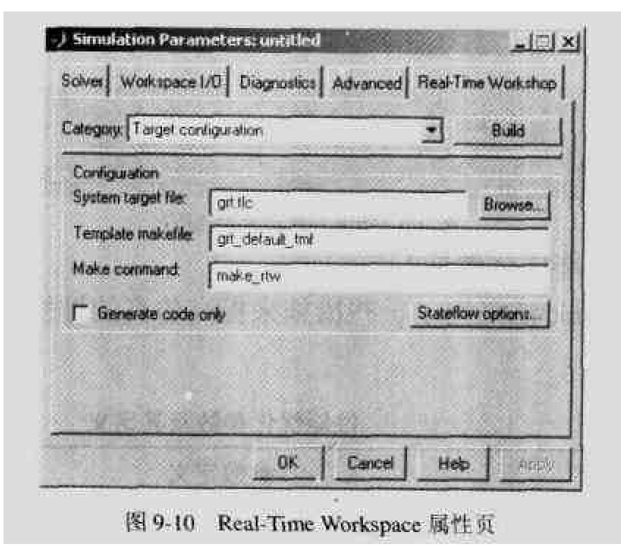



图 9-10 Real-Time Workspace 属性页

实时仿真（Real-Time Workshop）是对 MATLAB 和 Simulink 仿真能力的一种扩充，通过 Real-Time Workshop 我们可以方便地根据仿真模型生成能够在实时操作系统上运行的应用软件。关于 Real-Time Workshop 的内容已经超出了本书的范围，在此不作详细讨论。

### 9.1.2 运行仿真

在 MATLAB 中可以通过两种方式运行仿真：图形方式和命令行方式。在图形方式中，用户首先通过 Simulink 打开所需的仿真模型，然后从菜单栏或工具栏中选择相应的命令启动或终止仿真。在命令行方式中，用户通过在 MATLAB 工作区手工输入相应的命令来控制仿真的过程。这两种方式具有相同的效果，但是前者比较直观，操作方便，后者则便于通过代码来控制仿真程序的运行，尤其是需要在改变某些参数之后多次重复执行仿真的场合。

#### 1. 图形方式

在图形方式中，首先在 Simulink 中打开所需运行的仿真模块，选择菜单栏上的 Simulation

| Start 或者单击工具栏上的按钮，启动仿真程序。图 9-11 所示时启动仿真程序之后的仿真界面。这时候在窗口的下方显示了仿真的进度，同时可以通过选择菜单栏上的 Simulation | Pause（或单击工具栏上的按钮）暂停仿真，或者是通过选择菜单栏上的 Simulation | Stop 命令（或单击工具栏上的按钮）停止仿真。

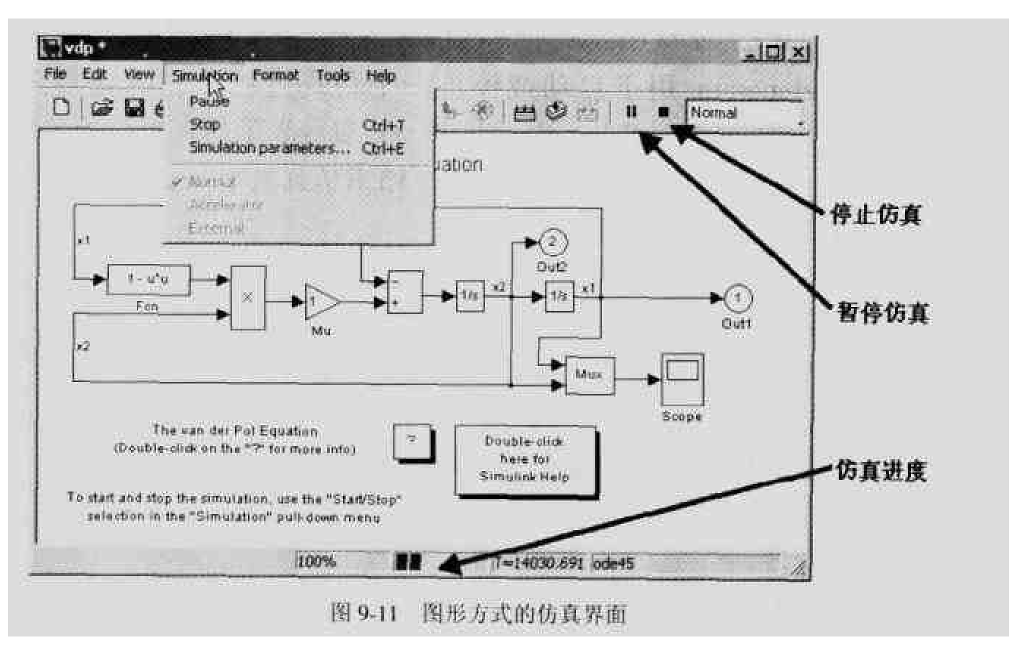



图 9-11　图形方式的仿真界面

用户可以在仿真过程中修改某些仿真参数，如仿真的结束时间，仿真求解器的类型，以及最大仿真步长等。同时，用户还可以通过选中某个信号线在浮动示波器或浮动显示模块中观察该信号线中的数据。如果用户对仿真模块参数的修改不会影响仿真过程的状态数、抽样间隔、过零检测数、仿真模块的向量长度或者模块内部工作向量的长度，这时候用户还可以在不中断仿真进程的条件下修改模块的参数设置。但是，仿真模块的结构（即模块的组成及其连接关系）在仿真过程中是不可以改变的。一般情况下，如果需要对仿真模块做出比较大的修改，应该先停止仿真，在修改完各种参数值之后再重新运行仿真程序。

### 2. 命令行方式

除了上一小节介绍的图形方式之外，我们还可以在 MATLAB 工作区中输入 sim 命令行启动仿真程序。完整的命令行格式是[*t*, *x*, *y*] = sim(model, timespan, options, *ut*)，其中 *t* 是仿真的时间向量，*x* 是仿真模块的内部状态，*y* 是输出向量，model 是仿真模块的名字，timespan 是仿真的开始时间和结束时间，options 是仿真的其他参数设置，*ut* 是一些附加参数。对于没有在命令行中表示的参数，Simulink 将根据仿真参数对话框（选择菜单栏的 Simulation | Simulation parameters…打开仿真参数对话框）的设置执行仿真。

MATLAB 函数 sim 的参数 options 是一个结构，它提供了若干仿真参数设置，包括仿真求解器名称和误差容限等。结构 options 需要通过 simset()函数来创建。例如，对于表 9-11 所示的仿真模块，可以在 MATLAB 工作区中以如下 3 种方式来启动仿真程序：

```
>>% 只带一个参数的命令行
```

```
>>[t,x,y] = sim('vdp')

>>% 通过 simset 设置参数 Refine 的数值
>>[t,x,y] = sim('vdp', [], simset('Refine',2));

>>% 设置仿真模块的运行时间为 1000 秒
>>% 通过 simset 设置参数 MaxRows、OutputVariables 和 FinalStateName 的数值
>>[t,x,y] = sim('vdp', 1000, simset('MaxRows', 100, 'OutputVariables', 'ty', 'FinalStateName', 'xFinal'));
```

MATLAB 函数 set_param 用于启动或停止一个仿真过程，它的完整命令行格式是 set_param('sys','SimulationCommand','cmd')，其中 sys 参数表示仿真模块的名称，cmd 参数表示所需执行的命令，包括'start'（开始仿真）、'stop'（结束仿真）、'pause'（暂停仿真）、'continue'（继续仿真）和'update'（更新仿真参数）。

相应地，MATLAB 函数 get_param 返回仿真过程的当前状态，它的完整命令行格式是 get_param('sys','SimulationStatus')，其中 sys 参数表示仿真模块的名称。需要注意的是，参数 SimulationStatus 是一个字符串。函数 get_param 的返回值等于'stopped'（仿真结束）、'initializing'（正在初始化）、'running'（正在运行）、'paused'（仿真暂停）、'updating'（正在更新仿真参数）、'terminating'（正在停止仿真）或'external'（用于 Real-Time Workshop 仿真）。

## 9.2 调试和分析

在设计仿真模型的过程中可能会引起各种错误，这些错误将导致仿真模型不能够启动运行，或者是运行之后得到与设计目标不一致的结果。一些错误是可以通过检查仿真模块排除的，对于一些比较隐秘的错误，Simulink 提供了一个集成的仿真调试环境，使得用户可以深入到仿真的每一个步骤去排除可能出现的错误。同时，通过 Simulink 用户可以方便地显示、保存和分析仿真数据，协助用户完成仿真的后期工作。

### 9.2.1 调试仿真模型

Simulink 提供了一个仿真调试器，用于对仿真模型可能存在的错误进行定位和跟踪。在仿真调试状态下，用户既可以以单步运行的方式执行仿真，也可以设置一些断点以方便地观察各个模块的输入信号和输出信号。总而言之，仿真调试器是一个功能强大的排错和纠错工具，是仿真过程中不可缺少的帮手。

**1. 打开调试窗口**

在 Simulink 打开仿真模型，选择菜单栏上的 Tools | Simulation debugger...，这时候出现如图 9-12 所示的仿真调试器。

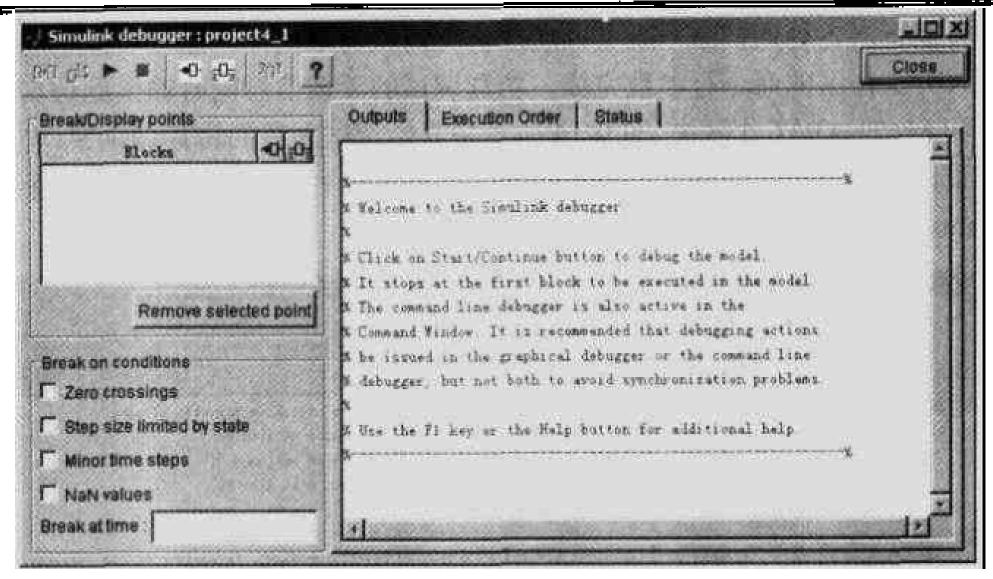



图 9-12　仿真调试窗口的运行界面

另外，还可以通过在 MATLAB 工作区中输入 sldebug 命令或在 sim 命令中添加 debug 参数打开指定仿真模型的仿真调试器。下面程序段中所列的两种命令行方式都能够打开仿真调试器。

```
>>% 通过 sldebug 命令打开仿真调试器
>>sldebug('project4_1');
>>
>>% 通过 sim 命令打开仿真调试器
>>sim('project4_1',[0,10],simset('debug','on'));
```

2. **执行仿真调试**

要运行仿真调试，单击仿真调试窗口中的开始执行按钮▶，Simulink 开始以调试方式执行仿真模型，这时候 Simulink 停留在仿真模型的第一个子模块上，如图 9-13 所示。图中标成黄色的模块是正在等待执行的子模块，并且左边的列表显示出这个子模块在整个模块中的位置。

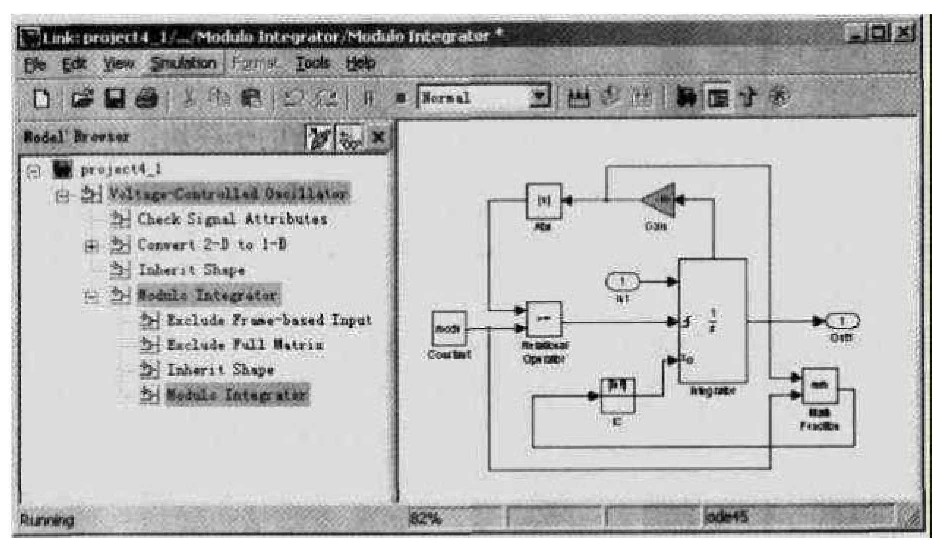



图 9-13　Simulink 的调试流程

在图形方式下，各种调试命令既可以通过仿真调试窗口执行，也可以在启动调试器之后通过 MATLAB 工作区的命令行参数完成，仿真调试的反馈信息同时输出到仿真调试窗口的 Output 面板和 MATLAB 工作区中。图 9-14 所示是开始调试之后的仿真调试窗口，其中 Output 面板中显示了调试进程的反馈信息，Execution Order 面板列出了仿真模型中各个子模块的执行顺序，Status 面板则显示了关于仿真模型的系统信息。

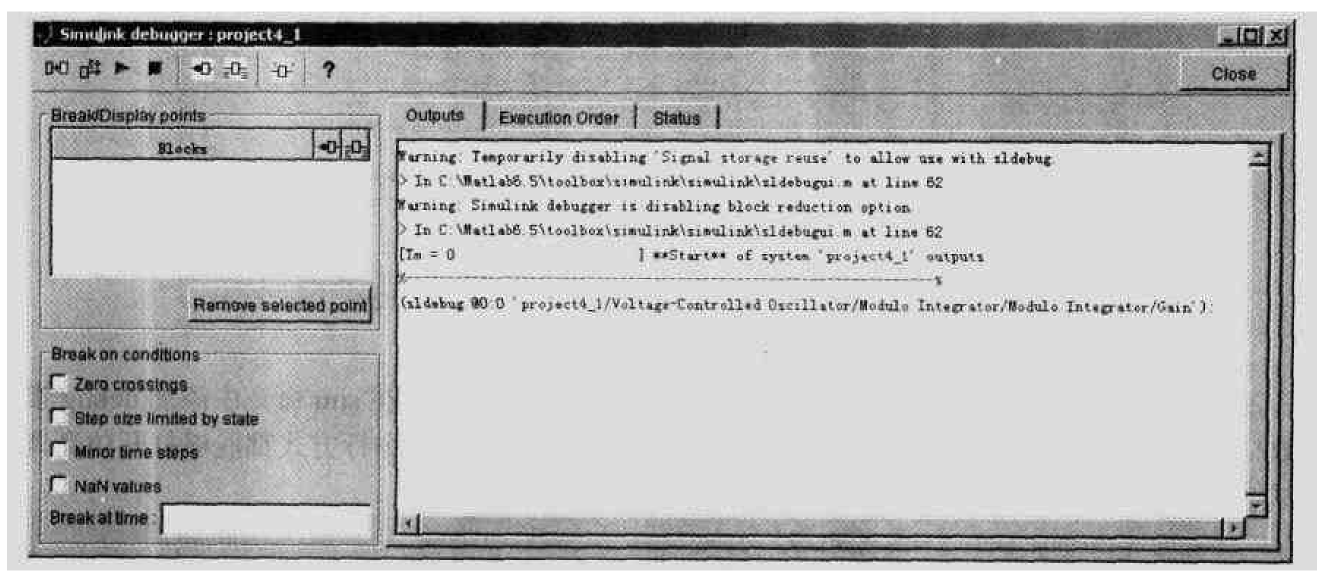



图 9-14 开始调试之后的仿真调试窗口

■ 继续执行仿真调试

当仿真调试窗口的停止仿真按钮有效时表示 Simulink 的调试过程暂时停止，正在等待用户的下一个调试命令。要继续执行调试，单击按钮，Simulink 运行仿真程序。如果仿真调试过程中没有设置任何断点（关于断点设置的方法将在下一节介绍），Simulink 将运行到仿真过程结束。

在命令行方式下，如果需要 Simulink 执行完仿真流程的剩余部分，可以在 MATLAB 工作区中输入命令 run，这时候 Simulink 将忽略仿真模型中的所有断点。

■ 执行到下一个模块

在仿真调试窗口中单击按钮，Simulink 将执行当前的子模块，在 Output 窗口中显示关于这个子模块的输入信号和输出信号，然后停止在下一个将被执行的子模块上。下面的程序段是当 Simulink 执行了两个子模块之后 Output 面板的输出。在命令行方式中，这个功能是通过 step 命令实现的。

```
Warning: Simulink debugger is disabling block reduction option.
> In C:\MATLAB6.5\toolbox\simulink\simulink\sldebugui.m at line 62
[Tm = 0                    ] **Start** of system 'project4_1' outputs
%-----------------------------------------------------------%
(sldebug @0:0 'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Gain'):
U1 = [0]
```

```
Y1 = [0]
%--------------------------------------------------------------%
(sldebug @0:1 'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Abs'):
U1 = [0]
Zc =
NO]
Y1 = [0]
%--------------------------------------------------------------%
(sldebug @0:2 'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Constant'):
```

■　执行到下一个时刻

在仿真调试窗口中单击按钮，Simulink 将执行当前时刻所有剩余的子模块，在 Output 窗口中显示相关信息，然后停止在下一个时刻的第一个子模块上。下面的程序段是在 Simulink 中执行上述操作之后 Output 面板的输出。

```
[Tm = 1.427842125570794e-020 ] **Start** of system 'project4_1' outputs
%--------------------------------------------------------------%
(sldebug @0:0 'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Gain'):
[Tm = 1.004754572603833e-005 ] **Start** of system 'project4_1' outputs
%--------------------------------------------------------------%
(sldebug @0:0 'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Gain'):
[Tm = 6.028527435622994e-005 ] **Start** of system 'project4_1' outputs
%--------------------------------------------------------------%
(sldebug @0:0 'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Gain'):
```

### 3. 设置调试断点

在仿真模型的调试过程中，用户可以设置一些断点，使得 Simulink 在到达指定的模块或时间点之后自动停下来。这种方式对于已经知道出错位置的仿真尤为有效，因为 Simulink 可以忽略掉一些无关的模块和事件，快速地把仿真流程设置到指定的位置。

■　设置模块的调试断点

要设置断点，首先在仿真模型中选择所需的模块，然后单击仿真调试窗口的按钮，则相应的模块被加入到仿真调试窗口的断点列表中，如图 9-15 所示。

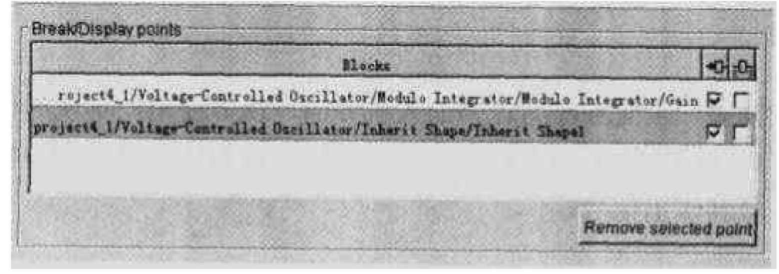



图 9-15　仿真调试窗口的断点列表

如果要取消某个断点，在断点列表中选中相应的模块，单击列表下方的 Remove selected point 按钮，则相应的模块将从断点列表中删除。

在命令行方式下可以通过命令 break 在指定模块的开始位置设定断点。break 可以有两种参数，一种是模块的名称，另一种是模块的编号。仿真调试窗口的 Execution Order 面板中列出了仿真模型各个子模块的名称和编号。另外，命令行方式还可以通过 bafter 命令在指定模块的结束位置设置断点。

■ 设置时间断点

Simulink 还允许用户设置仿真调试的时间断点。在图 9-16 所示的仿真调试窗口的 Break at time 输入框中输入相应的时间点（单位为秒），然后单击继续运行按钮，则仿真流程将停在大于这个设定的时间点的第一个抽样时刻。在命令行方式中，仿真调试的时间断点可以通过 tbreak 命令来完成。例如，在 MATLAB 工作区中输入命令 tbreak 9，则 Simulink 将在首次检测到仿真时间大于 9 秒时暂停执行。

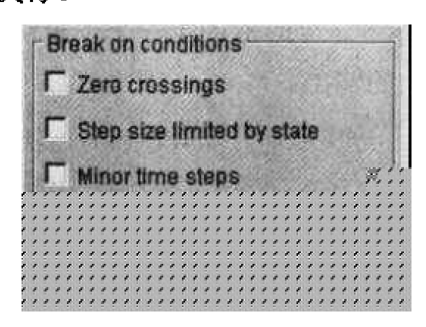



图 9-16　仿真调试窗口的中断条件

■ 设置 NaN 断点

选中图 9-16 所示中 NaN values 复选框，或者在 MATLAB 工作区中输入调试命令 nanbreak，则 Simulink 在计算过程中检测到无穷大或无穷小数值时（或者是数值超过了计算机的表示范围）将暂停仿真流程。

■ 设置步长断点

选中图 9-16 所示中与 Step size limited by state 复选框，或者在 MATLAB 工作区中输入仿真调试命令 xbreak，则 Simulink 在使用可变步长求解器的过程中检测到限制仿真步长的状态时将暂停仿真流程。步长断点一般用在需要耗费大量的仿真时间的仿真模型的调试过程中。

■ 设置过零点检测断点

选中图 9-16 所示中 Zero crossings option 复选框，或者在 MATLAB 工作区中输入调试命令 zcbreak，则 Simulink 在计算过程中检测到遗漏的过零点时将暂停仿真流程。这时候 Simulink 将显示这个过零点所在的模块以及这个过零点的类型。

#### 4. 显示调试信息

调试过程一般需要一个交互能力较强的人机界面，以方便用户实现对调试过程的控制。通过仿真调试窗口或 MATLAB 命令行，我们可以随时观察仿真模型的各种信息以及各个模块的状态、输入信号和输出信号。

■ 显示模块的输入输出信号

在仿真模型中选择所需的模块，然后单击仿真调试窗口中的按钮，则 Simulink 将把这个模块加入到仿真调试窗口的 Break/Display points 面板中，同时在执行到这个指定的模块时显示该模块的输入输出信号。命令行方式下的 trace 命令也能实现这个功能。

单击仿真调试窗口中的按钮，Simulink 将显示当前正在执行的模块的输入输出信号。相应地，通过 probe 命令能够在命令行方式中观察当前模块的输入输出信号。

■ 显示模块的执行顺序

选择仿真调试窗口中的 Execution Order 面板，或在命令行方式下输入 slist 命令可以观察仿真模型中各个模块的执行顺序。下面的程序段是 Execution Order 面板的内容，其中第一列表示模块的编号，第二列是模块的具体位置，第三列是模块的名称。

```
---- Sorted list for 'project4_1' [19 nonvirtual blocks, directFeed=0]
    0:0    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Gain' (Gain, tid=0)
    0:1    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Abs' (Abs, tid=0)
    0:2    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Constant' (Constant, tid=1)
    0:3    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Relational Operator' (RelationalOperator, tid=1)
    0:4    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Math Function' (Math, tid=0)
    0:5    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/IC' (InitialCondition, tid=0)
    0:6    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Modulo Integrator/Integrator' (Integrator, tid=0)
    0:7    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Inherit Shape/Inherit Shape1' (S-Function: scominhshape, tid=0,1)
    0:8    'project4_1/Voltage-Controlled Oscillator/sin' (Fcn, tid=0)
    0:9    'project4_1/Voltage-Controlled Oscillator/Inherit Shape/Inherit Shape1' (S-Function: scominhshape, tid=0,1)
    0:10   'project4_1/Scope' (Scope, tid=0)
    0:11   'project4_1/Pulse Generator' (DiscretePulseGenerator, tid=2)
    0:12   'project4_1/Voltage-Controlled Oscillator/Carrier frequency1' (Constant, tid=1)
    0:13   'project4_1/Voltage-Controlled Oscillator/Check Signal Attributes/Check Signal Attributes' (S-Function: sdspsigattrib, tid=1)
    0:14   'project4_1/Voltage-Controlled Oscillator/Convert 2-D to 1-D/Reshape' (S-Function: sreshape, tid=1)
    0:15   'project4_1/Voltage-Controlled Oscillator/Sensitivity' (Gain, tid=1)
    0:16   'project4_1/Voltage-Controlled Oscillator/Sum' (Sum, tid=1)
```

```
    0:17    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Exclude Frame-based Input /Check
Signal Attributes' (S-Function: sdspsigattrib, tid=1)
    0:18    'project4_1/Voltage-Controlled Oscillator/Modulo Integrator/Exclude Full Matrix/Check Signal
Attributes' (S-Function: sdspsigattrib, tid=1)
```

■ 显示仿真模型的状态

在仿真调试窗口中选择 Status 面板，或在命令行方式下输入 status 命令，可以观察仿真模型的当前状态。下面的程序段是 Status 面板的内容，其中列出了仿真的当前时间以及各种中断的设置情况。

```
%----------------------------------------------------------------%
  Current simulation time                          : 0 (MajorTimeStep)
  Default command to execute on return/enter : ""
  Stop in minor times steps                      : disabled
  Break at zero crossing events                  : disabled
  Break when step size is limiting by a state: disabled
  Time break point                                  : disabled
  Break on non-finite (NaN,Inf) values         : disabled
  Number of installed break points              : 796028788
[Tm = 0.2                        ] **Start** of system 'project4_1' outputs
```

## 9.2.2 分析仿真结果

在完成仿真模块的设计、运行和调试之后，就到了仿真的最后一个步骤：对仿真结果的处理和分析。相对于仿真运行的时间来说，仿真结果的分析是比较漫长的一个过程，因此，通常需要先把仿真数据保存起来，然后再对这些数据进行各种变换和处理，以获得所需的仿真结果。

### 1. 保存仿真数据

很多情况下，我们设计的仿真模型在结束仿真过程之后都会返回一些仿真数据，以验证设计的正确性和合理性。这些数据既可以直接通过 Simulink 模块直观地显示出来，也可以先保存到工作区或文件中，待仿真结束之后再作进一步的处理。一般说来，仿真模型返回地仿真数据通常有如下几种处理方式：

■ 直接显示

一般情况下，如果仿真结果不需要进行复杂的处理，可以把这些数据直接显示出来。Simulink 提供了多种模块用于显示仿真数据，比较常用的模块包括示波器模块（Scope）、浮动示波器模块（Floating Scope）、显示器模块（Display）、XY 图模块（XY Graph）。如果需要观察调制信号的特性，还可以使用连续时间眼图和发散图模块（Continuous-Time Eye and Scatter Diagrams）、离散时间眼图模块（Discrete-Time Eye Diagram Scope）、离散时间发散图模块（Discrete-Time Scatter Plot Scope）以及离散时间信号轨迹图模块（Discrete-Time Signal

Trajectory Scope)。另外，通过误差率统计模块（Error Rate Calculation）可以方便地观察信号的误比特率、误比特数以及输入信号总数。关于这些模块的用法请参考关于信源和信宿模块的相应内容。

■　保存到工作区

对于一些需要比较复杂的处理过程的仿真数据，或者需要保存起来作进一步处理的数据，可以把它们一方面通过各种显示模块直观地显示出来，另一方面保存到 MATLAB 工作区的变量中，供仿真结束后作进一步处理。Simulink 中的工作区写入模块（To Workspace）允许用户把仿真数据以数组或结构的形式保存到指定的工作区变量中。同时，通过工作区读取模块（From Workspace）还可以方便地从工作区变量中读取这些已经保存的数据。

另外，我们还可以把仿真模型中需要输出信号连接到输出模块（Out）中，然后在仿真参数中（在菜单栏中选择 Simulation | Simulation parameters...命令打开仿真参数设置对话框）设置输出信号在工作区中保存的变量名称。缺省情况下这些仿真数据保存在工作区中名为 *yout* 的变量中。

■　保存到文件

对于一些需要保持较长时间的仿真数据，应该把它们保存到文件中，以便于日后重新把这些数据读入到工作区中进行分析和处理。文件保存功能可以通过 Simulink 中的文件写入模块（To File）直接在仿真过程中完成，也可以先通过工作区写入模块（To Workspace）保存到工作区，在仿真结束之后再通过 save 命令把工作区变量保存到文件中。

### 2. 分析数据的合理性

仿真过程并不是一帆风顺的，即使仿真程序运行的过程中没有出现任何的警告或错误，得到仿真数据并不意味着仿真过程的结束，因为还要检查仿真数据是否是所期望的那样。

如果仿真结果偏差较大，首先要检查仿真模型的设计是否正确，如需要检查仿真模型中各个子模块的连接关系是否合理，子模块中的参数设置是否正确，很多情况下结果出现偏差可能是由于某个子模块的参数设置错误造成的。同时，还可以检查一下仿真模块的输入数据是否合理，信源产生的数据是否达到预定的要求，等等。

如果经过检查之后你仍然认为你的仿真模型的设计没有问题，那么很可能是你的仿真参数设置的不甚合理。这时候可以考虑增加仿真数据的输出采样数（在菜单栏中选择 Simulation | Simulation parameters...命令打开仿真参数设置对话框，修改 Solver 面板的 Refine factor 的数值），或者是选择其他的仿真求解器，然后重新运行仿真程序。

仿真是一个螺旋式上升的过程，在这个过程中可能需要对原来的仿真模型进行若干次的修改、重新设计和调试，才能实现最初的设计目标。

# 第 10 章　cdma 2000 移动通信系统

cdma 2000 由 IS-95（Interim Standard - 95）移动通信系统演进而来，它在室内环境中能够达到的最高传输速率为 2 Mbit/s，步行环境下能够达到 384 kbit/s，车载环境下则能达到 144 kbit/s。IS-2000 是采用 cdma 2000 技术的正式标准总称，它制定了 cdma 2000 系统中基站和移动台的工作规范。本章将按照 IS-2000 协议的要求介绍设计 cdma 2000 系统的仿真模型的过程。

## 10.1　cdma 2000 系统简介

1995 年美国电信工业协会（TIA，Telecommunications Industry Association）正式颁布的窄带 cdma（N-cdma）标准 IS-95A，并且在 1998 年发布了 IS-95A 的增强型版本 IS-95B。IS-95B 能够实现更高的数据传输速率，最大传输速率可以达到 115 kbit/s。IS-95A 和 IS-95B 统称为 IS-95，cdmaOne 则是基于 IS-95 标准的各种 cdma 产品的总称。中国联通早期建设的 cdma 网络采用 IS-95B 标准，然后逐步过渡到 cdma 2000。

cdma 2000 1x 是 cdma 2000 的第一阶段，它与 IS-95 一样占用 1.25MHz 带宽，最高理论传输速率能够达到 2Mbit/s，可支持 308 kbit/s 的数据业务。同时，cdma 2000 将在核心网络中采用分组交换技术，能够支持移动 IP 业务。在很长一段时间里，cdma 2000 3x 被看作是 cdma 2000 的第二阶段，它在前向信道中采用三载波调制方式，每个物理信道占用 3.75 MHz 的带宽资源，通过使用更宽的频谱资源来获得更高的速率传输数据。但是由于 cdma 2000 1x EV 的出现，现在看来 cdma 2000 3x 在很长一段时间内未必会被运营商采纳。

为进一步加强 cdma 1x 的竞争力，3G 标准化组织 3GPP2（3rd Generation Partnership Project 2）从 2000 年开始在 cdma 2000 1x 基础上制定增强技术，即 cdma 2000 1x EV 技术（EV 即 Evolution）。目前 cdma 2000 1x EV 的发展分为两个阶段，第一阶段称为 cdma 2000 1X EV-DO，第二阶段则称为 cdma 2000 1X EV-DV。

cdma 2000 1X EV-DO（DO 表示 Data Only）技术采用单独的物理信道来实现高速数据业务的传输。美国高通公司（Qualcomm）提出的 HDR（High Data Rate）技术能够在 1.25 MHz 的带宽内达到 650 kbit/s 的平均数据传输速率，峰值速率达到 2.4 Mbit/s，目前已经正式被 3GPP2 采纳为 cdma 2000 1X EV-DO 的惟一标准。cdma 2000 1x EV-DV（DV 表示 Data and Voice）则能够在同一个物理信道上同时实现话音业务和数据业务的传输，在 1.25 MHz 带宽内实现 4.8 Mbit/s 的数据传输速率，频谱效率高达 3.84 bit/s/Hz。

早在 2000 年 10 月，韩国 SKT 就推出了世界上第一个商用 cdma 2000 1x 系统，传输速率高达 150kbit/s，超过了 IMT-2000 规定的 144kbit/s 的传输标准。2001 年 4 月，LG 电信也推出了 cdma 2000 1x 服务。中国联通目前正在对已经建成的 IS-95B 网络进行升级，计划推

出 cdma 2000 的高速数据业务。

由于 cdma 2000 1x 系统的空中信道承载能力优于 GPRS，而且还能够有效地提高系统容量，因此 cdma 2000 1x 的用户将获得更快的互联网接入速度，即使在话务高峰阶段也能以较快的速率浏览网络。同时，cdma 2000 1x 系统完全兼容 cdmaOne，因此 cdma 2000 1x 用户可以在没有升级的 cdmaOne 网络中拨打电话，而 cdmaOne 用户也可以在升级后的网络中进行无缝漫游。

### 10.1.1　cdma 2000 1x 关键技术

cdma 2000 可以工作在多个频段中，其中北美频段的反向信道（即上行信道）采用 824~849MHz，前向信道（即下行信道）采用 869~894MHz，这些频段与 AMPS 以及 IS-95 系统的频段划分保持一致，从而允许运营商实现对网络的逐步升级。cdma 2000 1x 的前向信道和反向信道的码片速率为 1.2288 Mbit/s，这个速率远远低于欧洲的 Wcdma 标准。

cdma 2000 1x 对 IS-95A 和 IS-95B 系统具有后向兼容能力，支持重迭蜂窝网结构，在越区切换期间共享公共控制信道，并且支持 IS-95A 和 IS-95B 系统的信令标准及其话音业务。同时，cdma 2000 1x 大大增强了系统的性能和容量，这得益于如下一些关键技术的应用。

■　前向快速功率控制

IS-95 采用了反向功率控制，基站以一定的周期向移动台发出功率控制指令，通知移动台增加或降低当前发射功率，使得基站测量到的各个移动台的接收功率大致相等，从而能够提高系统的容量。cdma 2000 在采用反向功率控制技术的同时还使用了前向功率控制技术，即移动台测量收到前向业务信道的信噪比 $E_b / N_0$，并且把这个测量值与一个预先设定的门限值进行比较，然后根据比较结果向基站发出调整基站发射功率的指令。与反向功率控制类似，前向功率控制的最大速率可以达到 800bit/s。

■　前向快速寻呼信道

在 cdma 2000 中，基站可以通过快速寻呼信道向移动台发出指令，决定移动台是处于监听寻呼信道还是处于低功耗状态的睡眠状态。这样，移动台不必长时间连续监听前向寻呼信道，从而能够减少移动台的激活时间，节省移动台的功耗。

另外，通过前向快速寻呼信道，基站可以向移动台快速发出最近几分钟内的系统参数消息，促使移动台根据这个参数消息做出相应的处理。

■　前向链路发射分集技术

cdma 2000 1x 采用直接扩频发射分集技术，它有两种工作方式，即正交发射分集和空时扩展分集。正交发射分集采用不同的正交 Walsh 码对两个数据流进行扩频，然后通过两个发射天线发射给移动台；空时扩展分集则使用空间两根分离的天线发射信号，这些信号使用相同的 Walsh 码。使用前向链路发射分集技术可以减少基站的发射功率，增强信号的抗瑞利衰落性能，增大系统容量。

■　反向相干解调

在 IS-95 中，反向信道没有导频信号，数据是通过非相干解调得到的。cdma 2000 1x 增

加了反向导频信道，因此基站可以利用反向导频信道的扩频信号获得相干信号，实现相干解调。通过反向相干解调能够提高反向链路的性能，降低移动台的发射功率，提高系统容量。

■ Turbo 码

Turbo 码具有优异的纠错性能，适用于高速率传输中对译码时延要求不高的数据业务，并可降低对发射功率的要求，增加系统容量。在 cdma 2000 1x 中，Turbo 码仅用于前向补充信道和反向补充信道中。

Turbo 编码器由两个 RSC 编码器、交织器和删除器组成，其中每个 RSC 编码器产生两路校验位信号，这两个输出信号经删除复用后形成 Turbo 码。Turbo 译码器则由两个软判决译码器、交织器和去交织器构成，经对输入信号交替译码、软判决多轮译码、过零判决后得到译码输出。

■ 灵活的帧长

与 IS-95 不同，cdma 2000 1x 支持 5ms 帧、10ms 帧、20ms 帧、40ms 帧、80ms 帧以及 160ms 帧等多种帧长，不同类型信道可以使用不同的帧长。前向基本信道、前向专用控制信道、反向基本信道、反向专用控制信道采用 5ms 帧或 20ms 帧，前向补充信道、反向补充信道采用 20ms 帧、40ms 帧或 80ms 帧，话音信道则采用 20ms 帧。一般说来，较短的帧长度可以减少时延，但解调性能较差，而较大的帧长度则可以降低对发射功率要求，尤其适用于传输高速率的数据业务。

■ 增强的媒体接入控制功能

cdma 2000 的媒体接入控制子层控制多种业务接入，保证多媒体的实现。媒体接入控制子层支持话音业务、分组数据业务和电路数据业务，并且支持不同的服务质量(QoS)，与 IS-95 相比，可以满足更高传输速率和更多业务种类的要求。

### 10.1.2 cdma 2000 的信道划分

在介绍 cdma 2000 的信道划分之前，先介绍两个概念：扩频速率（Spreading Rate）和无线配置（Radio Configuration）。扩频速率是指 cdma 2000 信道调制中采用的载波数目。对于采用一个载波的调制方式（即占用 1.25MHz 的带宽）的 cdma 2000 系统，它的扩频速率等于 1(SR1)；而对于采用 3 个载波的调制方式，它占用了 3.75MHz 带宽，扩频速率等于 3(SR3)。一般说来，扩频速率 1 对应于 cdma 2000 1x，而扩频速率 3 则对应于 cdma 2000 3x。

在 cdma 2000 系统中，前向信道和反向信道可以采用多种传输速率和帧长，这些传输速率和帧长可以划分成不同的无线配置。cdma2000 前向信道共有 9 种无线配置方式，而反向信道则有 6 种不同的无线配置方式。其中，前向信道和反向信道的前两种无线配置(RC1 和 RC2)分别对应于 IS-95 中的两种传输速率。

图 10-1 所示是 cdma 2000 1x 反向 cdma 信道的分配情况。从图 10-1 所示中可以看到，在 cdma 2000 1x 系统中，反向信道由反向接入信道（RACH，Reverse Access Channel）、反向业务信道（TCH，Traffic Channel）、增强接入信道（EACH，Enhanced Access Channel）以及反向公共控制信道（CCCH，Common Control Channel）组成。

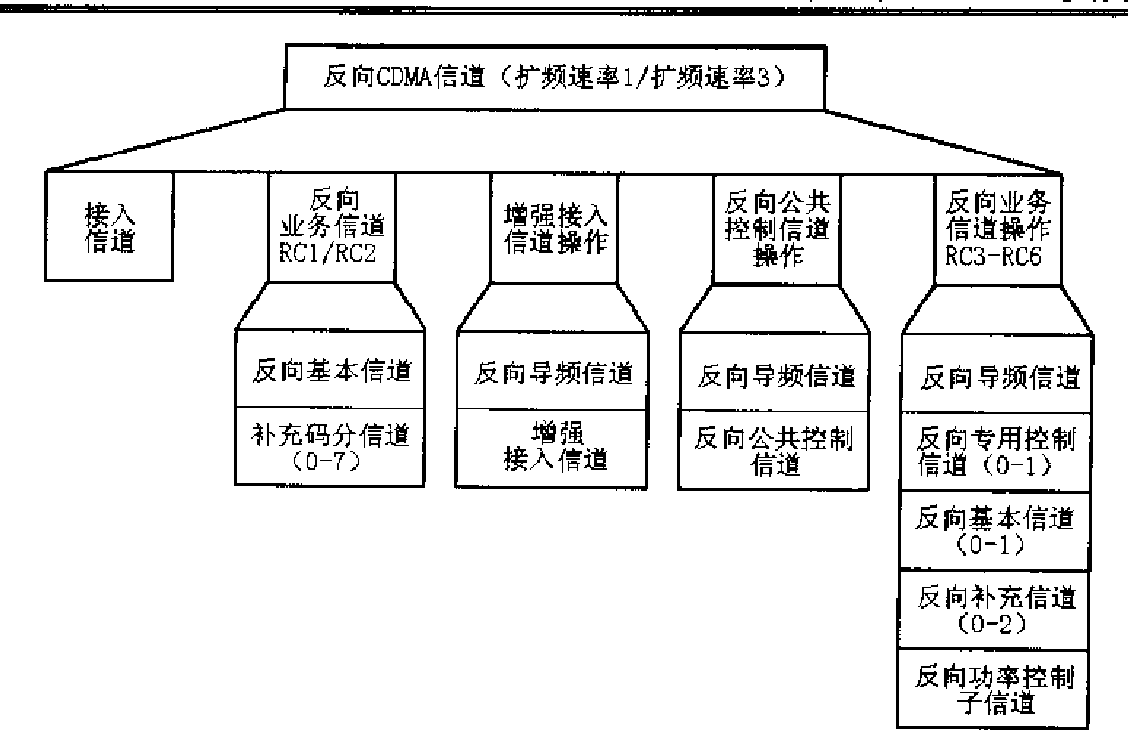



图 10-1　cdma 2000 1x 反向 cdma 信道

对于无线配置方式 1 和 2（RC1 和 RC2），反向业务信道由一条反向基本信道（FCH，Fundamental Channel）和 0~7 条反向补充码分信道（SCCH，Supplemental Code Channel）组成，这是对 IS-95A 和 IS-95B 系统的兼容。而在无线配置方式 3~6 中，反向业务信道包含了反向导频信道（PICH，Pilot Channel）、反向专用控制信道（DCCH，Dedicated Control Channel）、反向基本信道、反向补充信道（SCH，Supplemental Channel）以及反向功率控制子信道（Reverse Power Control Subchannel）。

cdma 2000 1x 前向 cdma 信道则比较复杂，它除了包含 IS-95 系统的各个前向信道之外，还增加了几种新的信道类型，如图 10-2 所示。

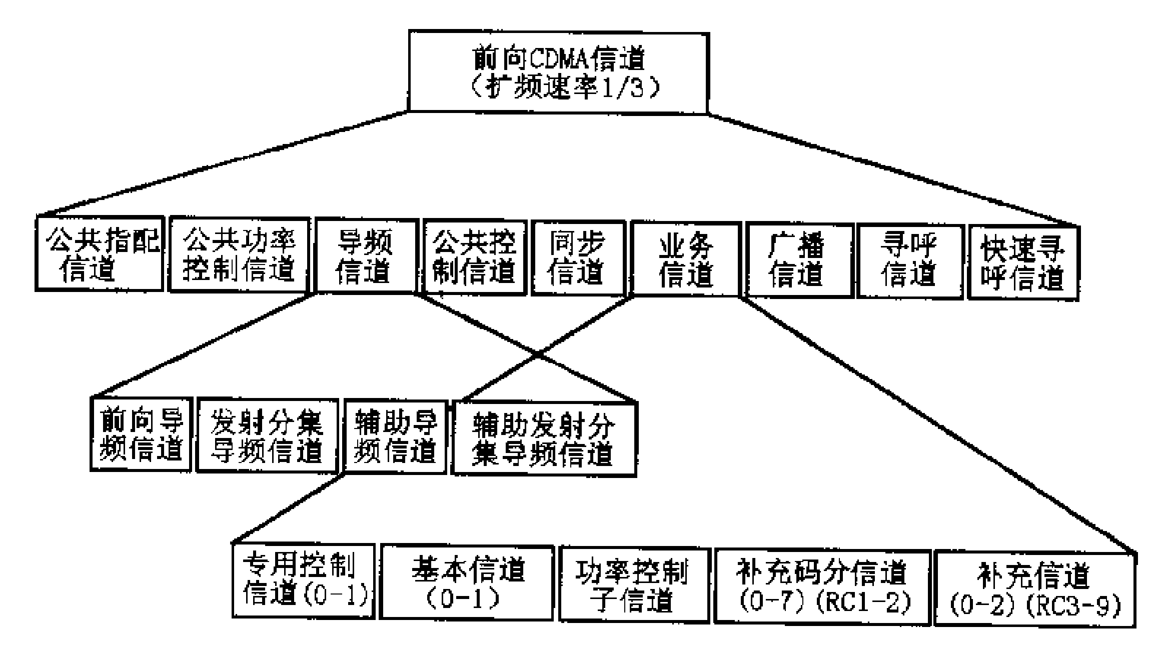



图 10-2　cdma 2000 1x 前向 cdma 信道

cdma 2000 1x 前向导频信道包含了 4 种类型的导频信道：前向导频信道（Forward Pilot Channel）、发射分集导频信道（TDPICH，Transmit Diversity Pilot Channel）、辅助导频信道（APICH，Auxiliary Pilot Channel）以及辅助发射分集导频信道（ATDPICH，Auxiliary Transmit Diversity Pilot Channel），基站根据不同的工作模式使用不同的导频信道。

与反向业务信道类似地，前向业务信道也包括前向基本信道（FCH，Fundamental Channel）、前向专用控制信道（DCCH，Dedicated Control Channel）、前向功率控制子信道（Forward Power Control Subchannel）。对于 RC1 和 RC2，前向业务信道可以有多至 7 条前向补充码分信道（SCCH，Supplemental Code Channel），而对于 RC3~9，前向业务信道可以包含 0~2 条前向补充信道（SCH，Supplemental Channel）。

cdma 2000 1x 保留了 IS-95 中的同步信道（Sync Channel）和寻呼信道（PACH，Paging Channel），分别用于为整个小区内的移动台提供同步信号和寻呼信息。另外，cdma 2000 1x 还增加了快速寻呼信道（QPCH，Quick Paging Channel），能够实现更快的寻呼功能。

除此之外，cdma 2000 1x 前向信道还包含公共指配信道（Common Assignment Channel）、公共功率控制信道（Common Power Control Channel）、公共控制信道（Common Control Channel）和广播信道（Broadcast Channel），分别用于实现业务信道的指配、广播信息的发布以及各种相关的控制功能。

本章将以 cdma 2000 扩频速率 1（SR1）为基础介绍 cdma 2000 前向信道和反向信道的建模和仿真过程。

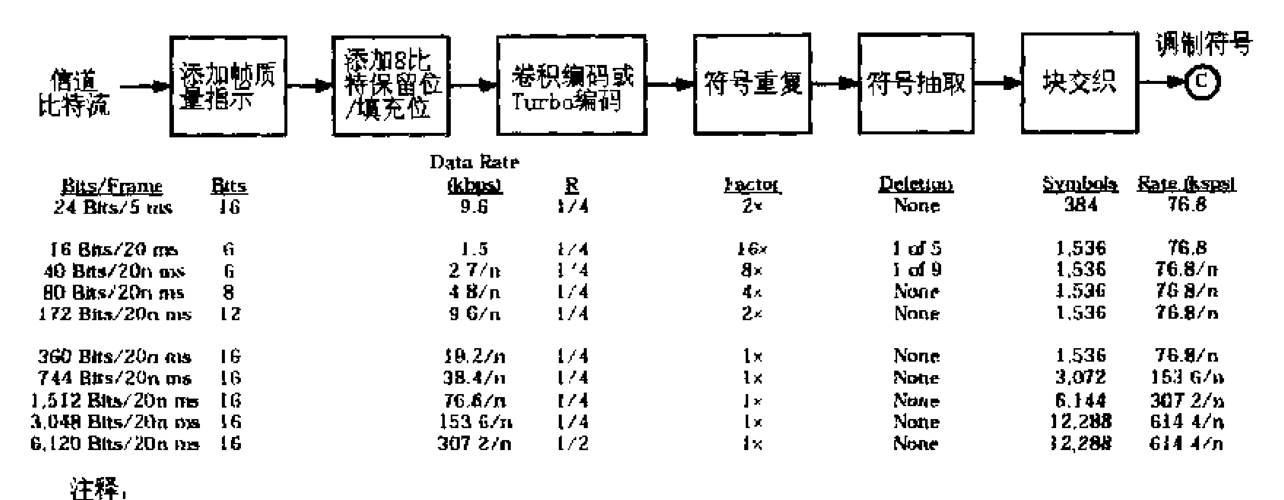
## 10.2 cdma 2000 反向业务信道

在 cdma 2000 中，反向业务信道包括反向基本信道、反向专用控制信道和反向补充信道（或者是 RC1 和 RC2 中反向补充码分信道），其中反向基本信道和反向专用控制信道主要用于传输速率较低的话音数据以及控制信令数据，而反向补充信道则可以实现高速率的数据传输。本节将介绍一个 cdma 2000 反向业务信道的仿真模型。

### 10.2.1 cdma 2000 反向业务信道简介

在 cdma 2000 中，单载波 cdma 2000（即 cdma 2000 1x）的反向业务信道使用无线配置 3 和 4（RC3 和 RC4），三载波 cdma 2000（cdma 2000 3x）则采用无线配置 5 和 6（RC5 和 RC6）。本章将只介绍反向业务信道 RC3 和 RC4。

cdma 2000 1x 反向业务信道数据帧在调制之前一般要经过 CRC 编码、卷积编码、信号重复、信号抽取以及块交织等过程。图 10-3 所示是无线配置 3（RC3）数据帧在调制之前的处理流程。




| Bits/Frame | Bits | Data Rate (kbps) | R | Factor | Deletion | Symbols | Rate (ksps) |
|---|---|---|---|---|---|---|---|
| 24 Bits/5 ms | 16 | 9.6 | 1/4 | 2× | None | 384 | 76.8 |
| 16 Bits/20 ms | 6 | 1.5 | 1/4 | 16× | 1 of 5 | 1,536 | 76.8 |
| 40 Bits/20n ms | 6 | 2.7/n | 1/4 | 8× | 1 of 9 | 1,536 | 76.8/n |
| 80 Bits/20n ms | 8 | 4.8/n | 1/4 | 4× | None | 1,536 | 76.8/n |
| 172 Bits/20n ms | 12 | 9.6/n | 1/4 | 2× | None | 1,536 | 76.8/n |
| 360 Bits/20n ms | 16 | 19.2/n | 1/4 | 1× | None | 1,536 | 76.8/n |
| 744 Bits/20n ms | 16 | 38.4/n | 1/4 | 1× | None | 3,072 | 153.6/n |
| 1,512 Bits/20n ms | 16 | 76.8/n | 1/4 | 1× | None | 6,144 | 307.2/n |
| 3,048 Bits/20n ms | 16 | 153.6/n | 1/4 | 1× | None | 12,288 | 614.4/n |
| 6,120 Bits/20n ms | 16 | 307.2/n | 1/2 | 1× | None | 12,288 | 614.4/n |

注释：
1、n是帧的持续时间（表示为20毫秒的整数倍）。对于40比特的帧，n = 1或2；对于长度大于40比特的帧，n = 1，2或4。
2、5毫秒帧只适用于反向基本信道；另外，反向基本信道只能使用n = 1时16至172比特长度的帧。
3、大于360比特的反向补充信道可以采用Turbo编码方式，除此之外只能采用卷积编码。
4、卷积编码需要8比特的填充位；Turbo编码8比特填充位的前两个比特是保留位，其余的6比特是Turbo编码器产生的数据。

图 10-3　RC3 反向基本信道和反向补充信道

从图 10-3 所示可以看到，RC3 支持多种类型的数据帧，这些数据帧的长度从 16bit 到 6120bit，帧长可以是 5ms，也可以是 20ms、40ms 或 80ms。每帧数据首先通过一个 CRC 编码器产生特定长度的 CRC 校验位，在 cdma 2000 中这些校验位被称为帧质量指示（FQI，Frame Quality Indicator）。同时，为了使卷积编码器在对一帧数据完成编码之后内部状态能够自动复位，RC3 在每个数据帧的末尾添加 8bit 的编码器尾部（Encoder Tail Bits，如果采用 Turbo 编码器则这 8bit 成为保留比特）。我们通常所说的数据传输速率是在完成上述两个过程之后得到的数据帧的传输速率，这其中包含了 CRC 编码产生的校验位和预留的 8bit 数据。

cdma 2000 1x 在 RC3 的反向业务帧中使用了码率为 1/4 和 1/2 的卷积编码器，这两种编码器的约束长度都等于 9。因此，数据帧在卷积编码之后长度分别增大到原来的 4 倍和 2 倍，通过提供足够的信息冗余度来提高信号的抗噪声性能。

RC3 数据帧通过卷积编码器之后，还需要根据数据传输速率决定是否对其实施信号重复，以及信号重复的倍数，从而把低速率的数据通过重复提高到较高的速率。对于特定长度的数据帧，cdma 2000 1x 还采用了抽取（Puncturing）技术，以一定的比例去掉卷积编码信号中的某些数据。接收端采用的卷积译码器能够根据抽取信号还原得到原来的数据，因此，信号抽取技术能够在不影响信号解码的条件下实现更高的数据传输速率。

最后，cdma 2000 1x 对每个反向业务帧实施交织（Interleaving）。通过交织，原先的数据传输顺序被打乱，数据帧中相邻的信息位在一定的时间间隔之后才被传输。接收端通过一个相反的变换（解交织）可以得到交织前的数据，从而能够有效地对抗由于信道衰落引起的突发传输错误。

对于 RC4 反向业务信道，数据帧在调制之前的处理过程也需要经过 CRC 编码、卷积编码、信号重复、信号抽取以及块交织等。同时，RC4 中某些长度的数据帧在进行 CRC 编码之前还需要添加 1bit 的保留比特（Reserved Bit），如图 10-4 所示。

信道比特流 → 添加预留位 → 添加帧质量指示 → 添加保留位/填充位 → 卷积编码/Turbo编码 → 符号重复 → 符号抽取 → 块交织 → 调制符号

| Bits/Frame | Bits | Bits | Data Rate (kbps) | R | Factor | Deletion | Symbols | Rate (ksps) |
|---|---|---|---|---|---|---|---|---|
| 24 Bits/5 ms | 0 | 16 | 9.6 | 1/4 | 2x | None | 384 | 76.8 |
| 21 Bits/20 ms | 1 | 6 | 1.8 | 1/4 | 16x | 8 of 24 | 1,536 | 76.8 |
| 55 Bits/20n ms | 1 | 8 | 3.6/n | 1/4 | 8x | 8 of 24 | 1,536 | 76.8/n |
| 125 Bits/20n ms | 1 | 10 | 7.2/n | 1/4 | 4x | 8 of 24 | 1,536 | 76.8/n |
| 267 Bits/20n ms | 1 | 12 | 14.4/n | 1/4 | 2x | 8 of 24 | 1,536 | 76.8/n |
| 552 Bits/20n ms | 0 | 16 | 28.8/n | 1/4 | 1x | 4 of 12 | 1,536 | 76.8/n |
| 1,128 Bits/20n ms | 0 | 16 | 57.6/n | 1/4 | 1x | 4 of 12 | 3,072 | 153.6/n |
| 2,280 Bits/20n ms | 0 | 16 | 115.2/n | 1/4 | 1x | 4 of 12 | 6,144 | 307.2/n |
| 4,584 Bits/20n ms | 0 | 16 | 230.4/n | 1/4 | 1x | 4 of 12 | 12,288 | 614.4/n |

注释：

1、n是帧的持续时间（表示为20毫秒的整数倍）。对于55比特的帧，n = 1或2，对于长度大于55比特的帧，n = 1，2或4。

2、5毫秒帧只适用于反向基本信道，另外，反向基本信道只能使用n = 1时21至267比特长度的帧。

3、大于552比特的反向补充信道可以采用Turbo编码方式，除此之外只能采用K = 9的卷积编码器。

4、卷积编码需要8比特的填充位，Turbo编码8比特填充位的前两个比特是保留位，其余的6比特是Turbo编码器产生的数据。

图 10-4　RC4 反向基本信道和反向补充信道

与 RC3 类似地，RC4 也支持 5ms 帧、20ms 帧、40ms 帧和 80ms 帧，但是这些数据帧的长度与 RC3 不同，从而产生不同的传输速率。每帧数据首先通过一个 CRC 编码器产生特定长度的 CRC 校验位，在 cdma 2000 中这些校验位被称为帧质量指示（FQI，Frame Quality Indicator）。同时，为了使卷积编码器在对一帧数据完成编码之后内部状态能够自动复位，RC3 在每个数据帧的末尾添加 8bit 的编码器尾部（Encoder Tail Bits，如果采用 Turbo 编码器则这 8bit 成为保留比特）。我们通常所说的数据传输速率是在完成上述两个过程之后得到的数据帧的传输速率，这其中包含了 CRC 编码产生的校验位和预留的 8bit 数据。

另外，对于长度较大的补充信道数据帧（RC3 长度不小于 360bit 的数据帧以及 RC4 长度不小于 552bit 的数据帧），除了采用卷积编码方式之外，还可以采用 Turbo 编码方式对数据帧进行编码。Turbo 编码把卷积和交织过程结合起来，能够增强信号的抗干扰能力，提高系统的性能。

在完成 cdma 2000 1x 反向业务数据帧的编码和交织之后，这些数据帧进入图 10-5 所示的调制器进行 QPSK 调制。

从图 10-5 所示中可以看出，反向业务信道数据帧的调制过程比较复杂。归纳起来，它涉及到 3 个过程：信道的正交化，PN 序列的产生以及正交调制。在 cdma 2000 1x 中，移动台可以同时获得多种类型的反向信道，包括反向基本信道、反向补充信道、反向导频信道、反向专用控制信道等，它们都在同一个物理信道中传输，占用相同的频率资源。cdma 2000 1x 采用了正交 Walsh 序列来实现信道的正交化，不同信道的数据采用不同的 Walsh 序列进行扩频，这些 Walsh 序列的码片速率都等于 1.2288Mchip/s。例如，假设 $d_1(t)$ 和 $d_2(t)$ 是两个信道在编码和交织后的数据流，它们的符号周期分别为 $T_1$ 和 $T_2$。另外，我们假设序列 $d_1(t)$ 和 $d_2(t)$ 分别采用 Walsh 序列 $W_i^n(t)$ 和 $W_j^n(t)$ 进行扩展，这两个 Walsh 序列的码片速率为 $T_c$，整个序列的周期为 $T_w = nT_c$，其中 $n = 2^m$，$m = 0,1,2,\cdots$。在扩频通信系统中，$T_1 >> T_c$，$T_2 >> T_c$。这两个信道在使用 Walsh 序列进行扩频之后，产生的输出信号 $s(t) = d_1(t)\cdot W_i^n(t) + d_2(t)\cdot W_j^n(t)$。我们知道，Walsh 序列的特点是这些序列具有很好的自相关特性和互相关特性，即：

$$\int_0^{T_w} W_i^n(t) W_j^n(t)\mathrm{d}t = \begin{cases} 0, i \neq j \\ T_w, i = j \end{cases}$$

因此，在接收端对 $d_1(t)$ 进行解调时，只要用 Walsh 序列 $W_i^n(t)$ 与接收信号 $s(t)$ 进行相乘和积分，即：

$$\int_0^{T_1} s(t)\cdot W_i^n(t)\mathrm{d}t=\int_0^{T_1}\mathrm{d}_1(t)\cdot W_i^n(t)\cdot W_i^n(t)\mathrm{d}t+\int_0^{T_1}\mathrm{d}_2(t)\cdot W_j^n(t)\cdot W_i^n(t)\mathrm{d}t,$$

其中第二项积分等于 0，从而得到关于第一个信道数据的积分输出。通过这种方式，cdma 系统能够把多个码分信道复用在一起，形成两个支路的输出信号 $d_I(t)$ 和 $d_Q(t)$。

关于 cdma 2000 1x 中 RC3 和 RC4 反向信道调制的第二个要点是它的 PN 序列（即伪随机噪声序列，Pseudo-random Noise Sequence）。cdma 2000 1x 通过一个长码产生器（Long Code Generator）产生一个码片速率为 1.2288Mchip/s、长度为 $2^{42}-1$ 的伪随机序列，这个伪随机序列与长度为 $2^{15}$ 的另外两个伪随机序列（I 支路和 Q 支路短码序列）进行变换之后得到用于扰码的两个序列 $p_I(t)$ 和 $p_Q(t)$。

最后，序列 $p_I(t)$ 和 $p_Q(t)$ 对反向业务信道两个支路的输出信号 $d_I(t)$ 和 $d_Q(t)$ 进行扰码，得到另外两个输出序列 $s_I(t)$ 和 $s_Q(t)$，其中：

$$\begin{cases} s_I(t)=p_I(t)d_I(t)-p_Q(t)d_Q(t)\\ s_Q(t)=p_Q(t)d_I(t)+p_I(t)d_Q(t)\end{cases}$$

这两个输出序列在经过基带滤波之后分别调制到载波频率 $f_c$ 上，形成射频输出信号。

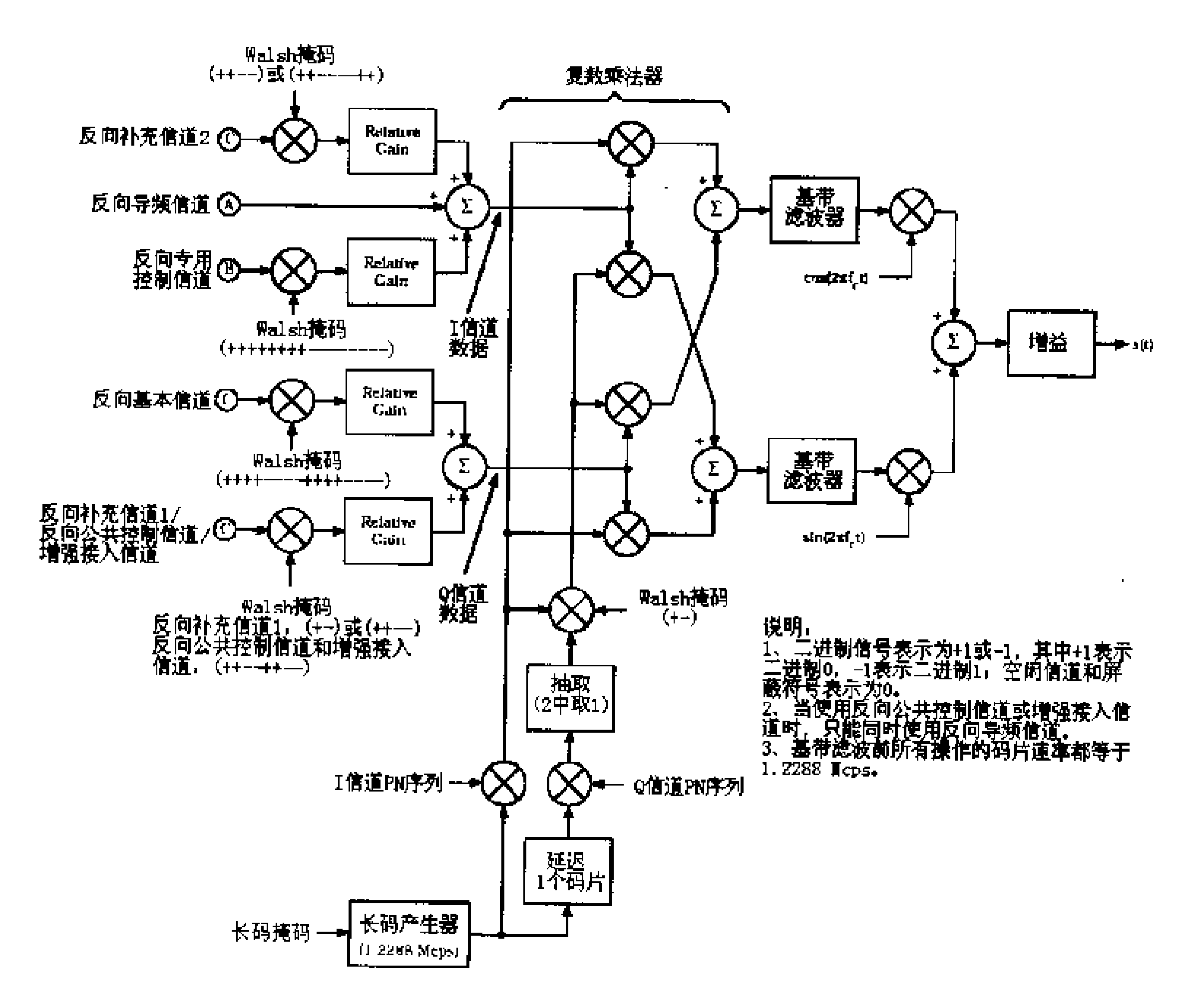



图 10-5　反向业务信道数据帧的调制过程

以上就是 cdma 2000 1x 系统中基站对 RC3 和 RC4 反向业务帧的处理过程。了解 IS-95 系统的读者可能已经注意到，这个过程与 IS-95 有不小的差别。作为对比，图 10-6 所示是 cdma 2000 系统对 RC1 反向业务数据帧的处理流程图。从图 10-6 所示中可以看到，IS-95 系统的调制方式比较简单。另外一个差别在于，在 IS-95 反向信道中，Walsh 序列用来实现对信号的 64 相正交调制（即每 6 个二进制数据位映射为 1 个 64bit 的 Walsh 序列），而不是像在 RC3 和 RC4 中那样用于区分不同的信道。

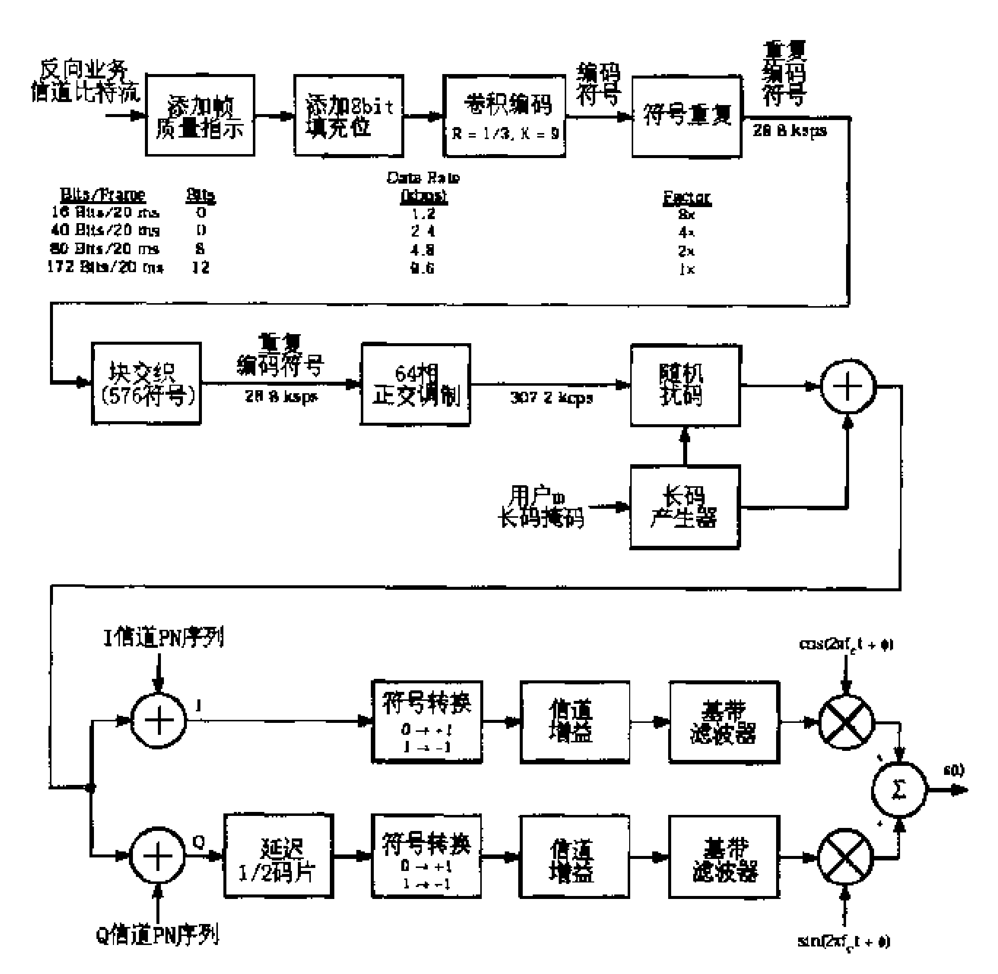



图 10-6　反向业务信道 RC1 的信号编码和调制

在后面的小节里，将按照 cdma 2000 协议规范实现 RC3 和 RC4 反向业务信道的编码、交织和调制过程。我们将按照模块化设计过程分别实现 cdma 2000 的 CRC 编码器模块、卷积编码器模块、信号交织器模块、正交扩频模块、PN 信号生成器模块和信号调制模块。此外，还设计了一个初始化模块，用于设置整个仿真模型中使用的一些公共参数。

## 10.2.2　CRC 编码器

在 cdma 2000 1x 中，每个数据帧首先经过一个 CRC 编码器，产生长度不等的 CRC 校验位（16bit、12bit、10bit、8bit 或 6bit）。这些 CRC 校验位称为帧质量指示（FQI，Frame Quality Indicator），它们用于协助接收端检查数据帧的好坏。

对于长度为 16bit 的 CRC 编码器，它的生成多项式为：

$$g(x)=x^{16}+x^{15}+x^{14}+x^{11}+x^{6}+x^{5}+x^{2}+x+1。$$

它适用于 RC3 中长度等于 24bit、360bit、744bit、1512bit、3048bit 和 6120bit 的数据帧，以及 RC4 中长度等于 24bit、552bit、1128bit、2280bit 和 4584bit 的数据帧。

对于长度为 16bit 的 CRC 编码器，它的生成多项式为：

$$g(x)=x^{12}+x^{11}+x^{10}+x^{9}+x^{8}+x^{4}+x+1。$$

它适用于 RC3 中长度等于 172bit 的数据帧，以及 RC4 中长度等于 267bit 的数据帧。

对于长度为 10bit 的 CRC 编码器，它的生成多项式为：

$$g(x)=x^{10}+x^{9}+x^{8}+x^{7}+x^{6}+x^{4}+x^{3}+1。$$

它用于计算 RC4 中长度等于 125bit 的数据帧的帧质量指示。

对于长度为 8bit 的 CRC 编码器，它的生成多项式为：

$$g(x)=x^{8}+x^{7}+x^{4}+x^{3}+x+1。$$

它适用于 RC3 中长度等于 80bit 的数据帧以及 RC4 中长度等于 55bit 的数据帧。

对于长度为 6bit 的 CRC 编码器，它的生成多项式为：

$$g(x)=x^{6}+x^{5}+x^{2}+x+1。$$

它适用于 RC3 中长度等于 16bit 和 40bit 的数据帧，以及 RC4 中长度等于 21bit 的数据帧。

另外，对于 RC1 和 RC2，cdma 2000 1x 还使用了另外一种长度为 6bit 的 CRC 编码器，其生成多项式为 $g(x)=x^{6}+x^{2}+x+1$。

需要注意的一点是，在对 RC4 中长度为 21bit、55bit、125bit 以及 267bit 的数据帧实施 CRC 编码之前，cdma 2000 1x 在这些数据帧之后添加一个 1bit 的填充位，使它们的长度分别变成 22bit、56bit、126bit 以及 268bit。

下面我们来设计 cdma 2000 移动台 CRC 编码器，这个 CRC 编码器能够根据信道的无线配置自动选择相应的 CRC 编码器，并且在输入的数据帧后面添加特定长度的帧质量指示（FQI）。图 10-7 所示是我们设计的 CRC 编码器的模块框图及其参数设置对话框。

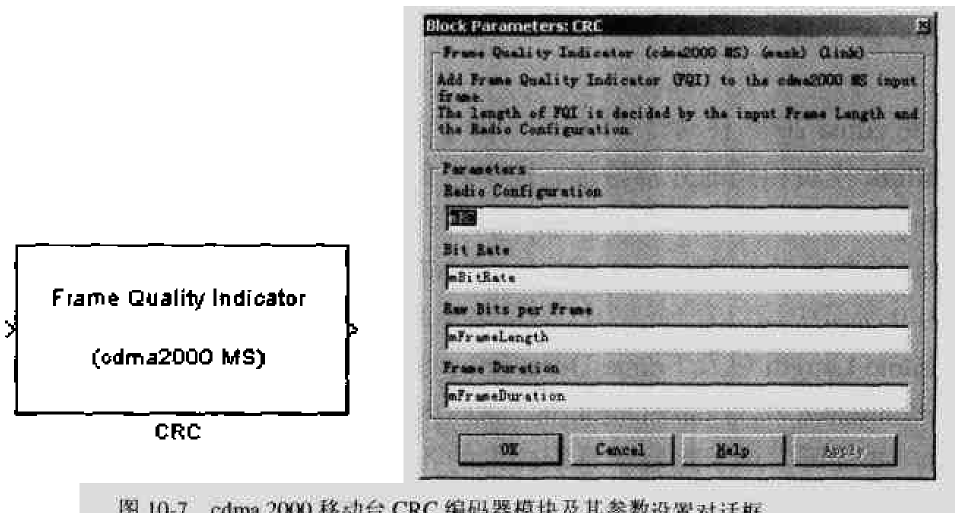



图 10-7　cdma 2000 移动台 CRC 编码器模块及其参数设置对话框

图 10-7 所示的 CRC 编码器是由 Simulink 模块库中的零填充(Zero Pad)模块和通用 CRC 编码器（General CRC Generator）模块构成的，如图 10-8 所示。

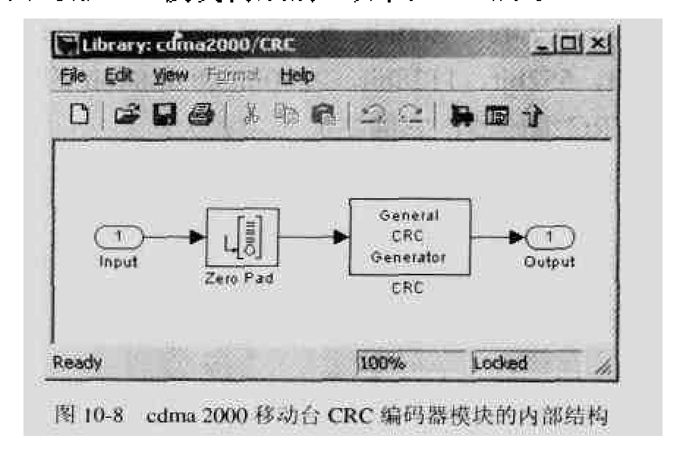



图 10-8 cdma 2000 移动台 CRC 编码器模块的内部结构

表 10-1 和表 10-2 列出了 cdma 2000 移动台 CRC 编码器模块中零填充模块和通用 CRC 器编码模块的参数设置情况，其中我们使用了两个变量 xPaddedFrameLen 和 xGenPoly，它们分别表示数据帧填充的长度以及 CRC 编码器的生成多项式。

**表 10-1　零填充（Zero Pad）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Zero Pad |
| Pad signal at | End |
| Pad along | Columns |
| Number of output rows | User-specified |
| Specified number of output rows | xPaddedFrameLen |
| Action when truncation occurs | None |

**表 10-2　通用 CRC 编码器（General CRC Generator）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | General CRC Generator |
| Generator polynomial | xGenPoly |
| Initial states | [0] |
| Checksums per frame | 1 |

在创建完 cdma 2000 移动台 CRC 编码器子系统之后，可以把这个子系统封装起来，使之具有与 Simulink 模块相同的外部特征。选中这个子系统，单击鼠标右键后从弹出式菜单中选择 Mask subsystem，然后选择 Edit mask 设置封装模块的外观和参数。图 10-9 所示是设置这个子系统的封装参数时的 Parameters 以及 Initialization 面板的属性。

从图 10-9 所示中可以看到，为这个封装子系统，创建了 4 个参数：Radio Configuration、Bit Rate、Frame Length 以及 Frame Duration，与这些参数对应的内部变量名称分别为 xRC、xBitRate、xFrameLen 和 xFrameDuration，这些内部变量对应于封转子系统中参数设置对话框的各个参数，并且可以用于封装子系统内部的各个模块中。

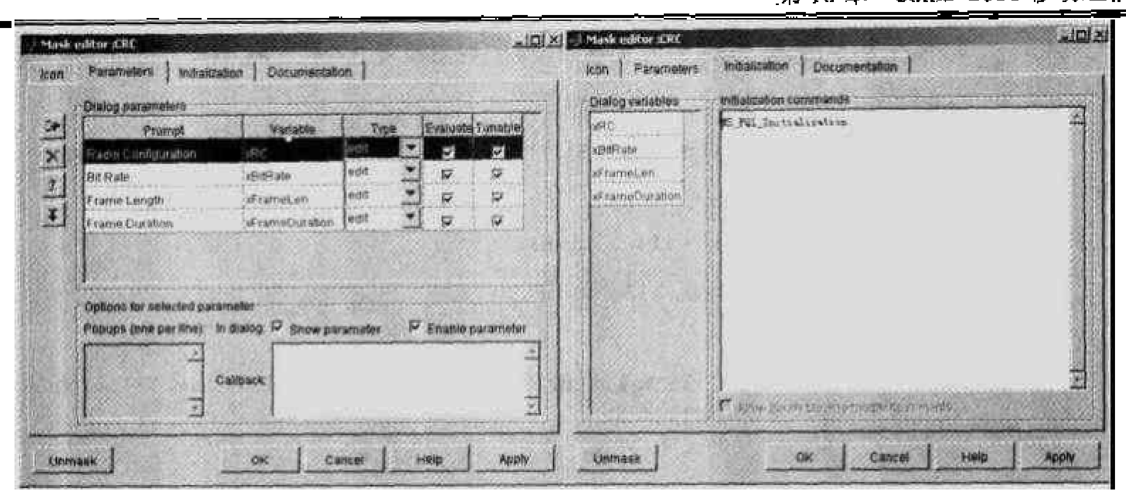


图 10-9　Parameters 和 Initialization 面板的参数设置

在 Initialization 面板中，通过 MS_FQI_Initialization.m 实现对上述各个参数的初始化过程。通过个单独的 M 文件来实现封装子系统的初始化使得用户能够方便地编写复杂的初始化代码，这种方式将应用于本章后面的各个子系统中。MS_FQI_Initialization.m 文件根据数据帧的无线配置情况和数据帧的长度选择相应的 CRC 编码生成多项式，其代码如下：

```
%================================================================
% MS_FQI_Initialization.m
%================================================================
% 大多数情况下不需要对数据帧进行填充，因此填充长度等于输入数据帧的长度
xPaddedFrameLength = xFrameLength;

% 根据无线配置设置 CRC 生成多项式和填充长度
% 只支持 RC3 和 RC4
switch xRC
    case 3
        % 根据输入数据帧的长度设置 CRC 生成多项式和填充长度
        switch xFrameLength
            % 长度为 16 和 40 的数据帧采用 6 位 CRC 编码
            case { 16, 40 }
                %g(x) = x6 + x2 + x + 1 for the 6-bit frame quality indicator (RC = 2)
                %g(x) = x6 + x5 + x2 + x + 1 for the 6-bit frame quality indicator (3 ≤ RC ≤ 6)
                xGenPoly = [6 2 1 0];
            % 长度为 80 的数据帧采用 6 位 CRC 编码
            case 80
                %g(x) = x8 + x7 + x4 + x3 + x + 1 for the 8-bit frame quality indicator
                xGenPoly = [8 7 4 3 1 0];
            % 长度为 172 的数据帧采用 6 位 CRC 编码
            case 172
                %g(x) = x12 + x11 + x10 + x9 + x8 + x4 + x + 1 for the 12-bit frame quality indicator
                xGenPoly = [12 11 10 9 8 4 1 0];
```

```
            % 长度为 24, 360, 744, 1512, 3048, 6120 的数据帧采用 6 位 CRC 编码
            case { 24, 360, 744, 1512, 3048, 6120 }
                %g(x) = x16 + x15 + x14 + x11 + x6 + x5 + x2 + x + 1 for the 16-bit frame quality
indicator
                xGenPoly = [16 15 14 11 6 5 2 1 0];
            % 如果输入数据帧的长度非法则报错
            otherwise
                error('Error: Invalid Frame Length for Radio Configuration 3 in cdma 2000 Mobile Station
block <Frame Quality Indicator>');
        end
    case 4
        % 根据输入数据帧的长度设置 CRC 生成多项式和填充长度
        switch xFrameLength
            % 长度为 21 的数据帧采用 6 位 CRC 编码，并且产生一个填充比特
            case 21
                xPaddedFrameLength = 22;
                %g(x) = x6 + x5 + x2 + x + 1 for the 6-bit frame quality indicator (3 ≤ RC ≤ 6)
                xGenPoly = [6 2 1 0];
            % 长度为 55 的数据帧采用 8 位 CRC 编码，并且产生一个填充比特
            case 55
                xPaddedFrameLength = 56;
                %g(x) = x8 + x7 + x4 + x3 + x + 1 for the 8-bit frame quality indicator
                xGenPoly = [8 7 4 3 1 0];
            % 长度为 125 的数据帧采用 10 位 CRC 编码，并且产生一个填充比特
            case 125
                xPaddedFrameLength = 126;
              %g(x) = x10 + x9 + x8 + x7 + x6 + x4 + x3 + 1 for the 10-bit frame quality indicator
                xGenPoly = [10 9 8 7 6 4 3 0];
            % 长度为 267 的数据帧采用 12 位 CRC 编码，并且产生一个填充比特
            case 267
                xPaddedFrameLength = 268;
                %g(x) = x12 + x11 + x10 + x9 + x8 + x4 + x + 1 for the 12-bit frame quality indicator
                xGenPoly = [12 11 10 9 8 4 1 0];
            % 长度为 24, 552, 1128, 2280, 4584 的数据帧采用 6 位 CRC 编码
            case { 24, 552, 1128, 2280, 4584 }
                %g(x) = x16 + x15 + x14 + x11 + x6 + x5 + x2 + x + 1 for the 16-bit frame quality
indicator
                xGenPoly = [16 15 14 11 6 5 2 1 0];
            % 如果输入数据帧的长度非法则报错
            otherwise
```

```
        error('Error: Invalid Frame Length for Radio Configuration 4 in cdma 2000 Mobile Station block <Frame Quality Indicator>');
            end
        % 如果输入数据帧的无线配置非法则报错
        otherwise
            error('Error: Invalid or Unhandled Radio Configuration in cdma 2000 Mobile Station block <Frame Quality Indicator>');
    end
```

在这个模块中，由于在封装窗口的 Initialization 面板中把 Initialization commands 设置为 MS_FQI_Initialization，因此 Simulink 在对模块实施初始化时将自动执行这个 M 文件的代码，从而实现对 xPaddedFrameLen 和 xGenPoly 参数的初始化。

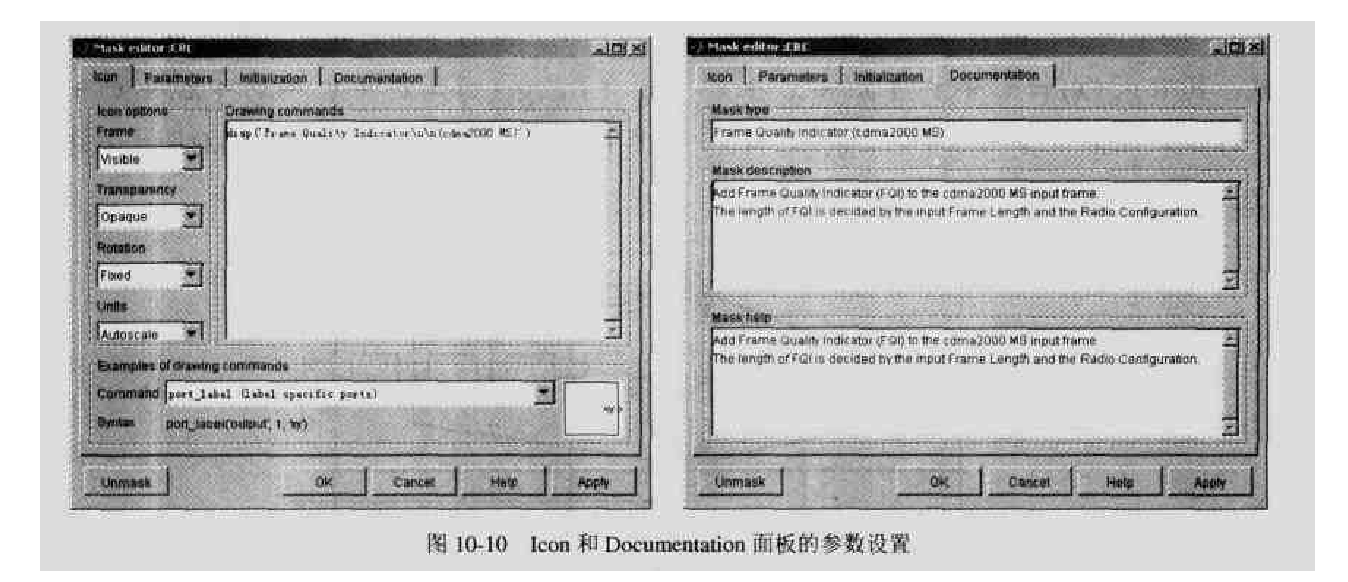



图 10-10　Icon 和 Documentation 面板的参数设置

最后，可以在封装参数设置窗口的 Icon 和 Documentation 面板中设置封装子系统的图标和说明文档信息，如图 10-10 所示。在 Icon 面板中，通过命令 disp('Frame Quality Indicator\n\n(cdma 2000 MS)')在模块中显示文字，使之具有易于识别的标志。在 Documentation 面板中，可以设置封装子系统的标题栏、说明栏以及帮助信息，其中前两项将出现在封装子系统的参数设置对话框中。在本章后面的章节中，将采用同样的方式设置封装子系统的 Icon 面板和 Documentation 面板。为了节省篇幅，将不再说明这两个面板的参数设置过程。

## 10.2.3　卷积编码器

cdma 2000 系统设计了 3 种类型的卷积编码器，它们的约束长度都等于 9，码率分别等于 1/4、1/3 和 1/2。图 10-11 所示是码率为 1/4 的卷积编码器的结构框图。

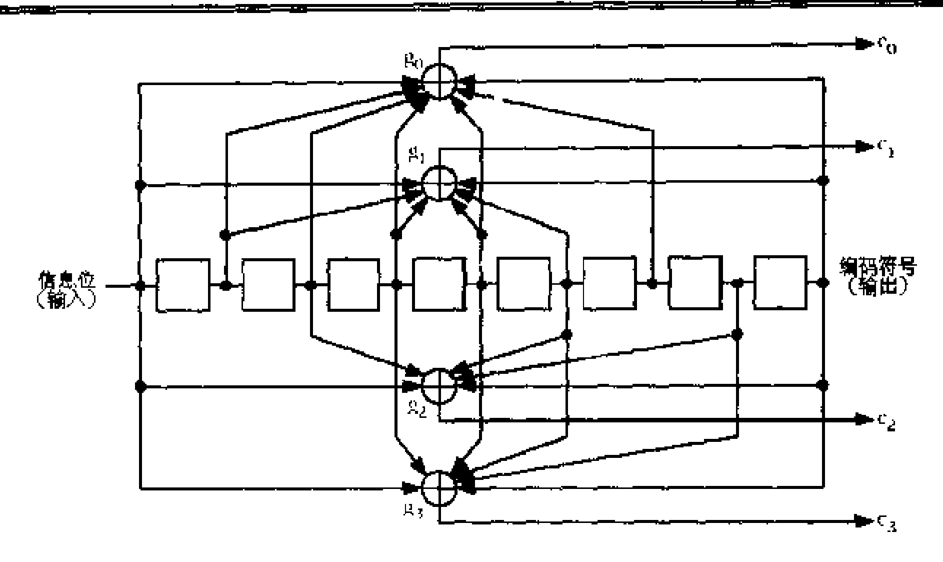



图 10-11 1/4 码率的卷积编码器

1/4 码率的卷积编码器产生的输出序列的长度是输入信号序列的 4 倍，它有 4 个生成多项式，其中 $g_0(x)=x^8+x^7+x^6+x^5+x^4+x^2+1$，$g_1(x)=x^8+x^7+x^5+x^4+x^3+1$，$g_2(x)=x^8+x^6+x^3+x+1$，$g_3(x)=x^8+x^5+x^4+x^3+x+1$，它们分别对应于八进制数 765、671、513 和 473。

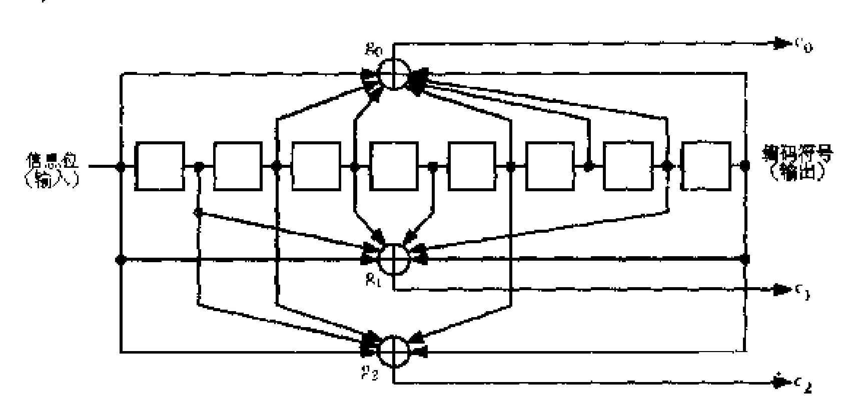



图 10-12 1/3 码率的卷积编码器

图 10-12 所示是码率等于 1/3 的卷积编码器，它产生的输出序列的长度是输入信号序列的 3 倍，并且这 3 个生成多项式分别为 $g_0(x)=x^8+x^6+x^5+x^3+x^2+x+1$，$g_1(x)=x^8+x^7+x^5+x^4+x+1$，$g_2(x)=x^8+x^7+x^6+x^3+1$，它们分别对应于八进制数 557、663 和 711。

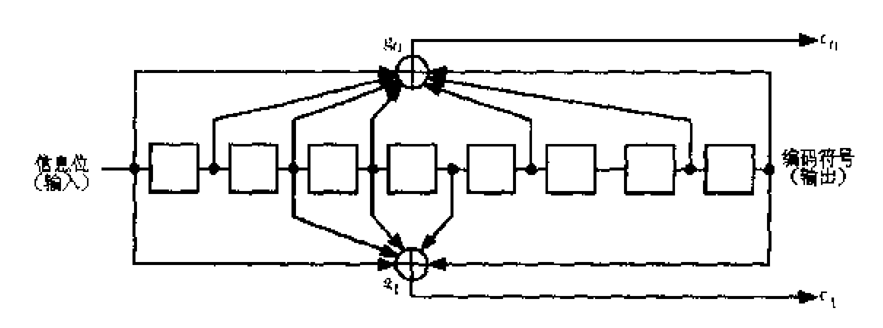



图 10-13 1/2 码率的卷积编码器

图 10-13 所示则是码率等于 1/2 的卷积编码器，它只有两个生成多项式，这两个生成多项式是 $g_0(x)=x^8+x^7+x^6+x^5+x^3+x+1$，$g_1(x)=x^8+x^6+x^5+x^4+1$，用八进制数表示为 753 和 561。

下面开始设计 cdma 2000 移动台的卷积编码器模块。在这个模块中，卷积编码器使用的生成多项式取决于输入数据帧的无线配置类型以及数据帧的长度。同时，卷积编码器还在每帧输入数据的后边预先添加 8 个 0，用于在每帧数据的编码结束之后对各个寄存器进行复位。cdma 2000 移动台卷积编码器模块框图及其参数设置对话框如图 10-14 所示。

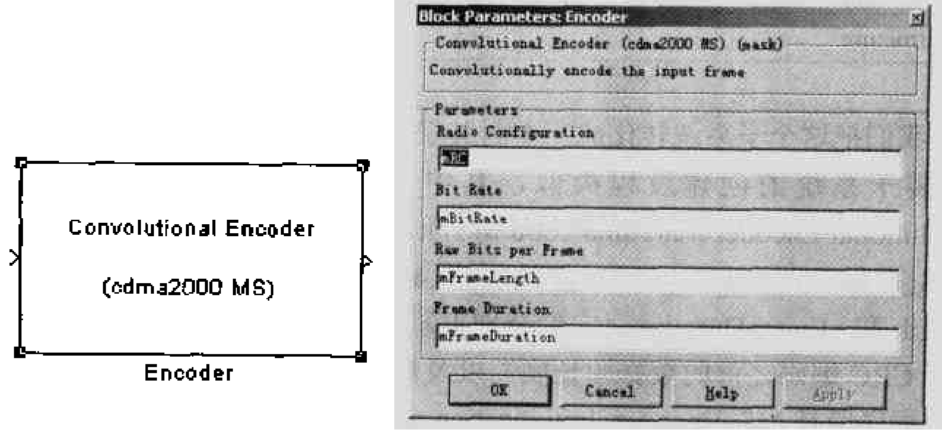



图 10-14 cdma 2000 移动台卷积编码器模块及其参数设置对话框

cdma 2000 移动台卷积编码器模块由一个零填充（Zero Pad）模块和一个卷积编码器（Convolutional Encoder）模块组成，这两个模块都是 Simulink 模块库中的自带模块。图 10-15 所示是 cdma 2000 移动台卷积编码器模块的组成框图。

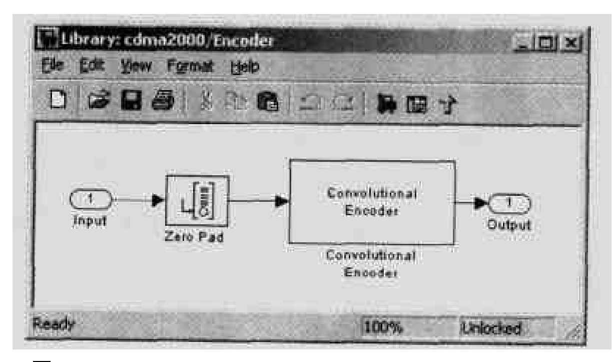



图 10-15 cdma 2000 移动台卷积编码器模块的内部结构

表 10-3 和表 10-4 分别列出了 cdma 2000 移动台卷积编码器模块中零填充模块和卷积编码器模块的参数设置情况，其中使用了两个封装子系统变量 xPaddedFrameLength 和 xTrellisStructure，它们分别表示添加编码尾部后数据帧的长度以及卷积编码器的 Trellis 参数设置。

**表 10-3　零填充（Zero Pad）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Zero Pad |
| Pad signal at | End |
| Pad along | Columns |
| Number of output rows | User-specified |

续表

| 参数名称 | 参数值 |
|---|---|
| Specified number of output rows | xPaddedFrameLength |
| Action when truncation occurs | None |

表 10-4　卷积编码器（Convolutional Encoder）模块的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Convolutional Encoder |
| Trellis structure | xTrellisStructure |
| Reset | None |

最后，我们把这个子系统转化成一个封装子系统。子系统的封装过程与上节介绍的 CRC 编码器封装子系统的创建过程类似，其中 Icon 面板的 Drawing commands 设置为 disp('Convolutional Encoder\n\n(cdma 2000 MS)');，Initialization 面板的 Initialization commands 设置为 MS_Encoder_Initialization，并且创建 4 个封装子系统内部变量 xRC、xBitRate、xFrameLength 和 xFrameDuration，它们分别表示输入数据帧的无线配置、数据率、原始数据帧长度（即 CRC 编码之前的数据帧长度）以及数据帧周期（单位是 ms）。

下面的程序段是 MS_Encoder_Initialization.m 文件的代码，其中实现了对封装子系统内部各个变量的初始化，这些代码将在 Simulink 初始化过程中执行。

```
%==========================================================================
% MS_Encoder_Initialization.m
%==========================================================================
% 根据无线配置设置卷积编码器 Trellis 参数和填充长度
% 只支持 RC3 和 RC4
switch xRC
    case 3
        % 根据输入数据帧的长度设置卷积编码器 Trellis 参数和填充长度
        switch xFrameLength
            case 6120
                xTrellisStructure = poly2trellis(9, [753 561]);
                xPaddedFrameLength = 6144;
            case 3048
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 3072;
            case 1512
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 1536;
            case 744
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 768;
            case 360
```

```
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 384;
            case 172
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 192;
            case 80
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 96;
            case 40
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 54;
            case 16
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 30;
            case 24
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 48;
            otherwise
                error('Error: Invalid Frame Length for Radio Configuration 3 in cdma 2000 Mobile Station
block <Convolutional Encoder>');
        end
    case 4
        % 根据输入数据帧的长度设置卷积编码器 Trellis 参数和填充长度
        switch xFrameLength
            case 4584
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 4608;
            case 2280
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 2304;
            case 1128
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 1152;
            case 552
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 576;
            case 267
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 288;
            case 125
```

```
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 144;
            case 55
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 72;
            case 21
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 36;
            case 24
                xTrellisStructure = poly2trellis(9, [765 671 513 473]);
                xPaddedFrameLength = 48;
            otherwise
                error('Error: Invalid Frame Length for Radio Configuration 4 in cdma 2000 Mobile Station
block <Convolutional Encoder>');
        end
    end
```

在使用 cdma 2000 移动台卷积编码器模块时，只需设置输入数据帧的无线配置、数据率、原始数据帧长度以及数据帧周期 4 个参数，模块将根据这 4 个参数的设置情况自动选择相应的卷积编码器，完成数据帧的卷积编码过程。

## 10.2.4 信号交织器

cdma 2000 移动台信号交织器模块实现对 RC3 和 RC4 反向业务帧的信号重复、信号抽取和信号交织功能，其模块框图和参数设置对话框如图 10-16 所示。

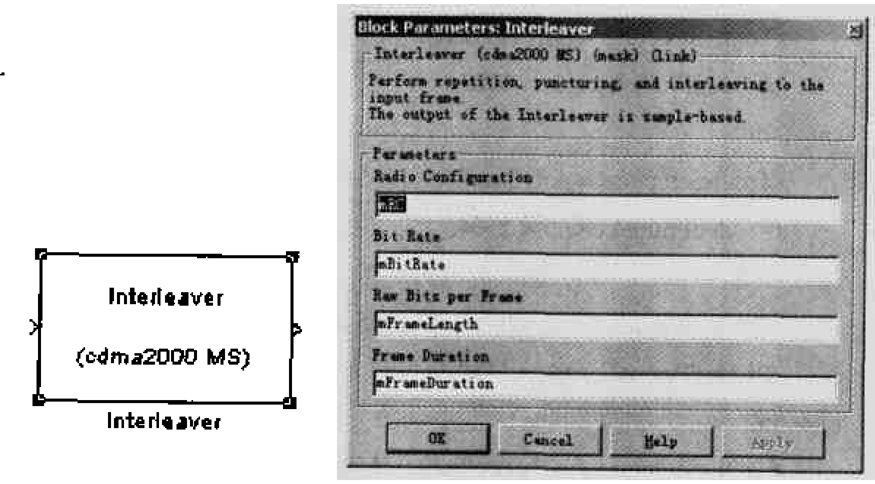



图 10-16　cdma 2000 移动台信号交织器模块及其参数设置对话框

图 10-17 所示是 cdma 2000 移动台信号交织器模块的内部结构，它由一个信号重复（Repeat）模块、一个信号抽取（Puncture）模块、两个帧状态转换（Frame Status Conversion 和 Frame Status Conversion1）模块和 S-函数（S-function）模块组成。

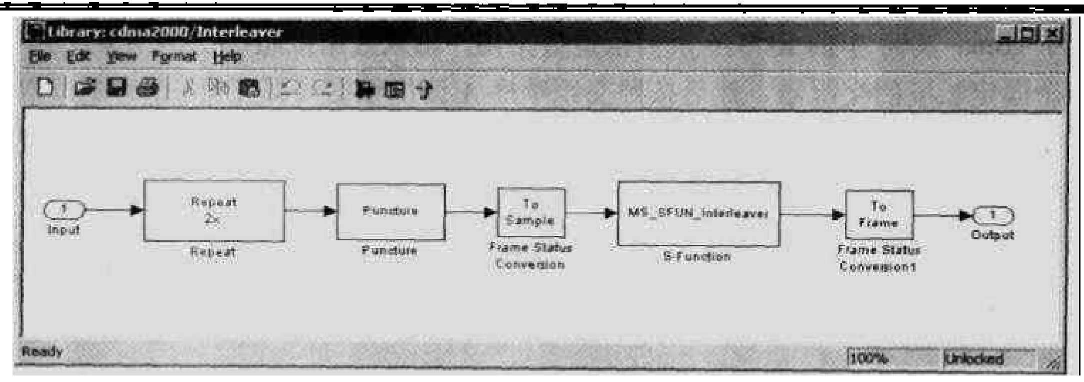



图 10-17 cdma 2000 移动台信号交织器模块的内部结构

表 10-5、表 10-6、表 10-7、表 10-8 和表 10-9 分别列出了 cdma 2000 移动台信号交织器模块中信号重复模块(Repeat)、信号抽取(Puncture)模块、帧状态转换(Frame Status Conversion 和 Frame Status Conversion1)模块和 S-函数(S-function)模块的参数设置情况。

**表 10-5　　信号重复(Repeat)模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Repeat |
| Repetition count | xRepetitionCount |
| Initial conditions | 0 |
| Frame-based mode | Maintain input frame rate |

**表 10-6　　信号抽取(Puncture)模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Puncture |
| Repetition count | xPunctureVector |

**表 10-7　　帧状态转换(Frame Status Conversion)模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Frame Status Conversion |
| Inherit output frame status from Ref input port | Unchecked |
| Output signal | Sample-based |

**表 10-8　　S-函数(S-function)模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | S-function |
| S-function name | MS_SFUN_Interleaver |
| S-function parameters | *xInterleaverSize* |

**表 10-9　　帧状态转换 1(Frame Status Conversion1)模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Frame Status Conversion |
| Inherit output frame status from Ref input port | Unchecked |
| Output signal | Frame-based |

对于 cdma 2000 中的 RC3~6 反向信道数据帧，输入数据按照 $0,1,2,\cdots,N-1$ 的下标顺序写入一个固定长度的交织器，然后按照交织顺序 $A_i$ 读出数据（即在第 $i$ 个时刻读取交织器中的第 $A_i$ 个数据），其中 $A_i = 2^m(i \bmod J) + BRO_m(\lfloor i/J \rfloor)$， $i=0,1,2,\cdots,N-1$， $BRO_m(y)$ 表示与 $m$ bit 整数 y 首尾翻转之后得到的二进制序列对应的整数。例如，6 表示成三位二进制数为 110，首尾翻转之后得到二进制序列 011，它对应于整数 3，因此， $BRO_3(6)=3$ 。在上述计算过程中，$m$ 和 $J$ 是由交织器长度决定的两个参数，它与交织器长度的对应关系如表 10-10 所示。

**表 10-10　　交织器参数**

| 交织器长度 | $m$ | $J$ |
|---|---|---|
| 384 | 6 | 6 |
| 768 | 6 | 12 |
| 1536 | 6 | 24 |
| 3072 | 6 | 48 |
| 6144 | 7 | 48 |
| 12288 | 7 | 96 |
| 576 | 5 | 18 |
| 2304 | 6 | 36 |
| 4608 | 7 | 36 |
| 9216 | 7 | 72 |
| 18432 | 8 | 72 |
| 36864 | 8 | 144 |

cdma 2000 移动台信号交织器模块的交织功能是由 M 文件 S-函数 MS_SFUN_Interleaver 实现的，它需要一个额外的输入变量，用于设置交织过程中使用的交织器的长度。S-函数根据这个输入参数计算数据帧的置换方式，并且输出交织后的数据帧。MS_SFUN_Interleaver.m 文件的代码如下：

```
function [sys,x0,str,ts] = MS_SFUN_Interleaver(t,x,u,flag, interleaver_size)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% M 文件 S-函数的主体部分
% 函数名称：MS_SFUN_Interleaver
% 主要功能：根据输入参数 flag 的数值调用相应的函数
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch flag,
  case 0,
    % 当 flag 等于 0 时调用 mdlInitializeSizes 函数执行初始化
    [sys,x0,str,ts]=mdlInitializeSizes(interleaver_size);
  case 2,
    % 当 flag 等于 2 时调用 mdlUpdate 函数计算离散状态的数值
    sys=mdlUpdate(t,x,u,interleaver_size);
  case 3,
```

```
        % 当 flag 等于 3 时调用 mdlOutputs 函数计算输出信号的数值
        sys=mdlOutputs(t,x,u);
    case {1, 4, 9}
        % 当 flag 等于 1、4 或 9 时没有相应的操作（没有连续状态）
        sys=[];
    otherwise
        % 当 flag 等于其他数值时表示仿真过程出错
        error(['Unhandled flag = ',num2str(flag)]);
end
% end MS_SFUN_Interleaver
%
%==============================================================================
% mdlInitializeSizes
% S-函数的初始化
% 向 Simulink 返回 S-函数各种信号的长度、初始设置和抽样时间设置
%==============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes(interleaver_size)
% 调用 simsizes 获得一个用于存放长度信息的结构
sizes = simsizes;
% 设置连续状态的个数
sizes.NumContStates = 0;
% 设置离散状态的个数
sizes.NumDiscStates = 2;
% 设置输出信号的个数为动态确定
sizes.NumOutputs = -1;
% 设置输入信号的个数为动态确定
sizes.NumInputs = -1;
% 设置直接反馈的状态：
%    0 表示没有直接反馈
%    1 表示存在直接反馈
sizes.DirFeedthrough = 1;
% 设置抽样时间的个数（大于等于 1）
sizes.NumSampleTimes = 1;
% 通过 simsizes 把 sizes 结构返回给 sys
sys = simsizes(sizes);
%
% 设置 S-函数的初始状态 x0
[m J]   = mdlInterleaverInit(interleaver_size);
x0 = [m J]';
%
```

```
% 设置 S-函数的保留参数 str（应该设置为空向量[]）
str = [];
%
% 初始化抽样时间
ts  = [-1 0];
% end mdlInitializeSizes
%
%==========================================================================
% mdlUpdate
% 更新 S-函数的离散状态并且向 Simulink 返回这些状态的数值
%==========================================================================
function sys=mdlUpdate(t,x,u,interleaver_size)
% 计算 S-函数的离散状态并且通过 sys 参数返回给 Simulink
[m J] = mdlInterleaverInit(interleaver_size);
sys = [m J]';
% end mdlUpdate
%
%==========================================================================
% mdlOutputs
% 计算 S-函数的输出信号并且返回给 Simulink 作为模块的输出
%==========================================================================
function sys=mdlOutputs(t,x,u)
% 计算 S-函数的输出信号并且通过 sys 参数返回给 Simulink
m = x(1);
J = x(2);
exp_m = 2^m;
A = zeros(exp_m*J, 1);
for i = 0:length(u)-1
    quotient = fix(i/J);
    BRO = 0;
    for k = 1:m
        if bitget(quotient, k) == 1
            BRO = bitset(BRO, m-k+1);
        end
    end
    A(exp_m*mod(i, J) + BRO + 1) = u(i+1);
end
sys = A;
% end mdlOutputs
%
```

```
%==========================================================================
% mdlInterleaverInit
% 初始化交织器的长度
%==========================================================================
function [m,J]=mdlInterleaverInit(interleaver_size)
switch interleaver_size
    case 384
        m = 6;
        J = 6;
    case 768
        m = 6;
        J = 12;
    case 1536
        m = 6;
        J = 24;
    case 3072
        m = 6;
        J = 48;
    case 6144
        m = 7;
        J = 48;
    case 12288
        m = 7;
        J = 96;
    case 576
        m = 5;
        J = 18;
    case 2304
        m = 6;
        J = 36;
    case 4608
        m = 7;
        J = 36;
    case 9216
        m = 7;
        J = 72;
    case 18432
        m = 8;
        J = 72;
    case 36864
```

```
            m = 8;
            J = 144;
        otherwise
            error('Error: Invalid Interleaver Size in MS_SFUN_Interleaver');
end
```

最后一个步骤是把这个子系统转化成一个封装子系统。在子系统的封装过程中，我们把Icon面板的Drawing commands设置为disp('Turbo Encoder\n\n(cdma 2000 MS)')，并且创建两个封装子系统内部变量xFrameLength和xCodeRate，它们分别表示输入信号的数据帧长度以及Turbo编码的速率。其中，xCodeRate是一个Popup方式的变量，它的选择项可以是rate 1/2、rate 1/3或rate 1/4。

下面的程序段是MS_Interleaver_Initialization.m文件的代码，其中实现了对封装子系统内部各个变量的初始化，这些代码将在Simulink初始化过程中执行。

```
%=====================================================================
% MS_Interleaver_Initialization.m
%=====================================================================
% 根据无线配置设置交织器的信号重复因子、信号抽取向量以及交织器长度
% 只支持RC3和RC4
switch xRC
    case 3
        % 根据原始数据帧的长度设置的信号重复因子、信号抽取向量以及交织器长度
        switch xFrameLength
            case 6120
                xRepetitionCount = 1;
                xPunctureVector = [1];
                xInterleaverSize = 12288;
            case 3048
                xRepetitionCount = 1;
                xPunctureVector = [1];
                xInterleaverSize = 12288;
            case 1512
                xRepetitionCount = 1;
                xPunctureVector = [1];
                xInterleaverSize = 6144;
            case 744
                xRepetitionCount = 1;
                xPunctureVector = [1];
                xInterleaverSize = 3072;
            case 360
                xRepetitionCount = 1;
                xPunctureVector = [1];
```

```
                xInterleaverSize = 1536;
            case 172
                xRepetitionCount = 2;
                xPunctureVector = [1];
                xInterleaverSize = 1536;
            case 80
                xRepetitionCount = 4;
                xPunctureVector = [1];
                xInterleaverSize = 1536;
            case 40
                xRepetitionCount = 8;
                xPunctureVector = [1 1 1 1 1 1 1 1 0];
                xInterleaverSize = 1536;
            case 16
                xRepetitionCount = 16;
                xPunctureVector = [1 1 1 1 0];
                xInterleaverSize = 1536;
            case 24
                xRepetitionCount = 2;
                xPunctureVector = [1];
                xInterleaverSize = 384;
            otherwise
                error('Error: Invalid Frame Length for Radio Configuration 3 in cdma 2000 Mobile Station
block <Interleaver>');
        end
    case 4
        % 根据原始数据帧的长度设置的信号重复因子、信号抽取向量以及交织器长度
        switch xFrameLength
            case 4584
                xRepetitionCount = 1;
                xPunctureVector = [1 1 0 1 1 0 0 1 1 0 1 1];
                xInterleaverSize = 12288;
            case 2280
                xRepetitionCount = 1;
                xPunctureVector = [1 1 0 1 1 0 0 1 1 0 1 1];
                xInterleaverSize = 6144;
            case 1128
                xRepetitionCount = 1;
                xPunctureVector = [1 1 0 1 1 0 0 1 1 0 1 1];
                xInterleaverSize = 3072;
```

```
            case 552
                xRepetitionCount = 1;
                xPunctureVector = [1 1 0 1 1 0 0 1 1 0 1 1];
                xInterleaverSize = 1536;
            case 267
                xRepetitionCount = 2;
                xPunctureVector = [1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0];
                xInterleaverSize = 1536;
            case 125
                xRepetitionCount = 4;
                xPunctureVector = [1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0];
                xInterleaverSize = 1536;
            case 55
                xRepetitionCount = 8;
                xPunctureVector = [1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0];
                xInterleaverSize = 1536;
            case 21
                xRepetitionCount = 16;
                xPunctureVector = [1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0];
                xInterleaverSize = 1536;
            case 24
                xRepetitionCount = 2;
                xPunctureVector = [1];
                xInterleaverSize = 384;
            otherwise
                error('Error: Invalid Frame Length for Radio Configuration 4 in cdma 2000 Mobile Station
block <Interleaver>');
        end
    end
```

cdma 2000 移动台交织器模块的参数设置方式与前面介绍的 CRC 编码器模块以及卷积编码器模块相同，只需设置输入数据帧的无线配置、数据率、原始数据帧长度以及数据帧周期 4 个参数。cdma 2000 移动台交织器模块将根据这 4 个参数的设置情况自动设置信号重复、信号抽取和信号交织参数，完成数据帧的交织过程。

## 10.2.5　正交扩频模块

cdma 2000 反向信道数据帧经过交织之后进入正交扩频模块。正交扩频模块根据信道的类型选择一个相应的 Walsh 码对这个输入数据进行扩频，这个 Walsh 码的码片速率等于 1.2288Mchip/s。由于不同的信道使用了不同的 Walsh 序列，因此移动台产生的不同信道的数

据可以通过同一个物理信道进行传输。图 10-18 所示是 cdma 2000 移动台正交扩频模块及其参数设置对话框。

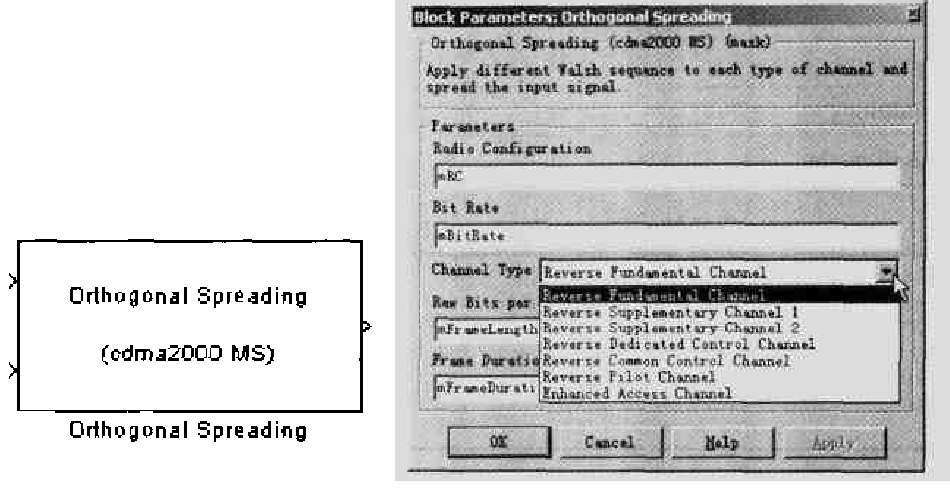



图 10-18 cdma 2000 移动台正交扩频模块及其参数设置对话框

由于 Walsh 码序列的码片速率（1.2288Mchip/s）明显高于各个信道交织后的数据传输速率，因此 cdma 2000 移动台正交扩频模块首先通过一个信号重复模块（Repeat）把输入信号的数据传输速率提高到 1.2288Mchip/s，然后通过极性转换器（Unipolar to Bipolar Converter）模块把二进制数据转换成双极性信号，再与 Hadamard 序列生成器（Hadamard Code Generator）模块产生的 Walsh 序列相乘，从而实现了正交扩频。cdma 2000 移动台正交扩频模块的内部结构如图 10-19 所示。

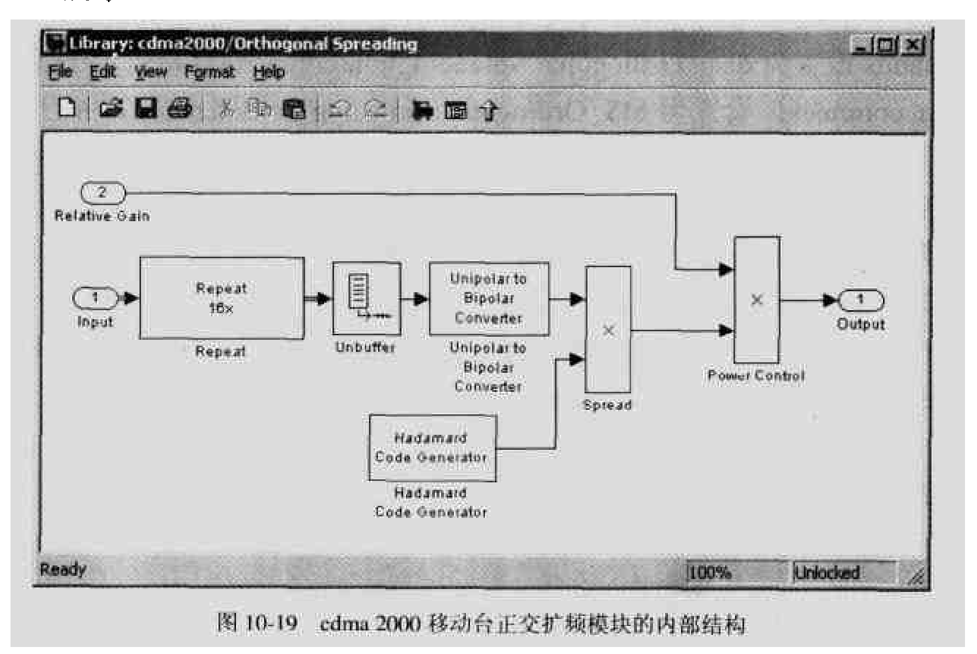



图 10-19 cdma 2000 移动台正交扩频模块的内部结构

另外，不同类型的信道具有不同的相对发射功率，因此，cdma 2000 移动台正交扩频模块的第二个输入端口用于输入信号发射功率的相对增益（绝对值）。表 10-11、表 10-12 和表 10-13 分别列出了信号重复模块、极性转换模块以及 Hadamard 序列生成模块的参数设置情况，其中变量 xRepetitionCount、xWalshCodeLength 和 xWalshCodeIndex 的数值将在模块初始化代码中进行定义。

表 10-11　　信号重复（Repeat）模块的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Repeat |
| Repetition count | xRepetitionCount |
| Initial conditions | 0 |
| Frame-based mode | Maintain input frame rate |

表 10-12　　极性转换器（Unipolar to Bipolar Converter）模块的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Unipolar to Bipolar Converter |
| M-ary number | 2 |
| Polarity | Negative |

表 10-13　　Hadamard 序列生成器（Hadamard Code Generator）模块的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Hadamard Code Generator |
| Code length | xWalshCodeLength |
| Code index | xWalshCodeIndex |
| Sample time | 1/1228800 |
| Frame-based outputs | Unchecked |

最后把这个子系统转化成一个封装子系统。其中，封装参数设置窗口中 Icon 面板的 Drawing commands 设置为 disp('Orthogonal Spreading\n\n(cdma 2000 MS)')，Initialization 面板的 Initialization commands 设置为 MS_OrthogonalSpreading_Initialization，并且创建五个封装子系统内部变量 xRC、xBitRate、xFrameLength、xFrameDuration 和 xChannelType，它们分别表示输入数据帧的无线配置、数据率、原始数据帧长度、数据帧周期以及信道的类型。

另外，参数 xChannelType 是一个 popup 类型的变量（即只能从下拉列表中选择参数的数值），它的选项是 Reverse Fundamental Channel、Reverse Supplementary Channel 1、Reverse Supplementary Channel 2、Reverse Dedicated Control Channel、Reverse Common Control Channel、Reverse Pilot Channel 或 Enhanced Access Channel，这些数值分别代表不同类型的信道。

下面的程序段是 MS_OrthogonalSpreading_Initialization.m 文件的代码，它根据信道的类型选择不同的 Walsh 码，并且根据数据帧的长度决定信号重复因子。

```
%=================================================================
% MS_OrthogonalSpreading_Initialization.m
%=================================================================
% 根据信道类型设置 Walsh 码的长度和序号
switch xChannelType
    case 1
        % 反向基本信道
        xWalshCodeLength = 16;
```

```
            xWalshCodeIndex = 4;
        case 2
            % 反向补充信道 1
            xWalshCodeLength = 2;
            xWalshCodeIndex = 1;
        case 3
            % 反向补充信道 2
            xWalshCodeLength = 4;
            xWalshCodeIndex = 2;
        case 4
            % 反向专用控制信道
            xWalshCodeLength = 16;
            xWalshCodeIndex = 8;
        case 5
            % 反向公共控制信道
            xWalshCodeLength = 8;
            xWalshCodeIndex = 2;
        case 6
            % 反向导频信道
            xWalshCodeLength = 32;
            xWalshCodeIndex = 0;
        case 7
            % 反向增强接入信道
            xWalshCodeLength = 8;
            xWalshCodeIndex = 2;
        otherwise
            error('Error: Invalid Channel Type for Radio Configuration 3 in cdma 2000 Mobile Station block
<Spreading and Modulation>');
    end

    % 根据无线配置设置输入信号的重复因子
    % 只支持 RC3 和 RC4
    switch xRC
        case 3
            switch xFrameLength
                case 6120
                    xRepetitionCount = fix((1228.8*xFrameDuration)/12288);
                case 3048
                    xRepetitionCount = fix((1228.8*xFrameDuration)/12288);
                case 1512
```

```
                    xRepetitionCount = fix((1228.8*xFrameDuration)/6144);
                case 744
                    xRepetitionCount = fix((1228.8*xFrameDuration)/3072);
                case 360
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 172
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 80
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 40
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 16
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 24
                    xRepetitionCount = fix((1228.8*xFrameDuration)/384);
                otherwise
                    error('Error: Invalid Frame Length for Radio Configuration 3 in cdma 2000 Mobile Station
block <Spreading and Modulation>');
            end
        case 4
            switch xFrameLength
                case 4584
                    xRepetitionCount = fix((1228.8*xFrameDuration)/12288);
                case 2280
                    xRepetitionCount = fix((1228.8*xFrameDuration)/6144);
                case 1128
                    xRepetitionCount = fix((1228.8*xFrameDuration)/3072);
                case 552
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 267
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 125
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 55
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 21
                    xRepetitionCount = fix((1228.8*xFrameDuration)/1536);
                case 24
                    xRepetitionCount = fix((1228.8*xFrameDuration)/384);
                otherwise
```

```
                error('Error: Invalid Frame Length for Radio Configuration 4 in cdma 2000 Mobile Station block <Spreading and Modulation>');
        end
    end
```

cdma 2000 移动台正交扩频模块的输出信号是一个抽样信号序列，信号的传输速率等于码片速率（1.2288Mchip/s）。不同信道的数据帧在通过各自的正交扩频模块之后叠加在一起，可以在同一个物理信道上进行传输，而接收端则可以通过这些信道使用的 Walsh 码把它们区分开来。

## 10.2.6　PN 信号生成器

cdma 2000 移动台 PN 信号生成器模块产生两个输出信号序列，分别用于对移动台产生的两个正交扩频信号进行扰码。图 10-20 所示是 cdma 2000 移动台 PN 信号生成器模块及其参数设置对话框。

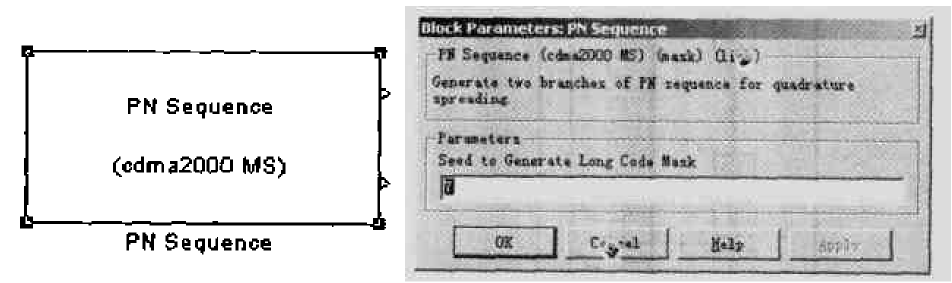



图 10-20　cdma 2000 移动台 PN 信号生成器模块及其参数设置对话框

在 cdma 2000 移动台 PN 信号生成器模块中，长码生成器（Long Code Generator）模块产生一个长度为 $2^{42}-1$ 的长码序列，序列的码片速率等于 1.2288Mchip/s。同时，I 支路 PN 序列生成（I branch PN）模块和 Q 支路 PN 序列生成（Q branch PN）模块分别产生一个长度为 $2^{15}$ 的短码序列，它们与长码序列进行变换之后就得到了两个输出 PN 序列，如图 10-21 所示。

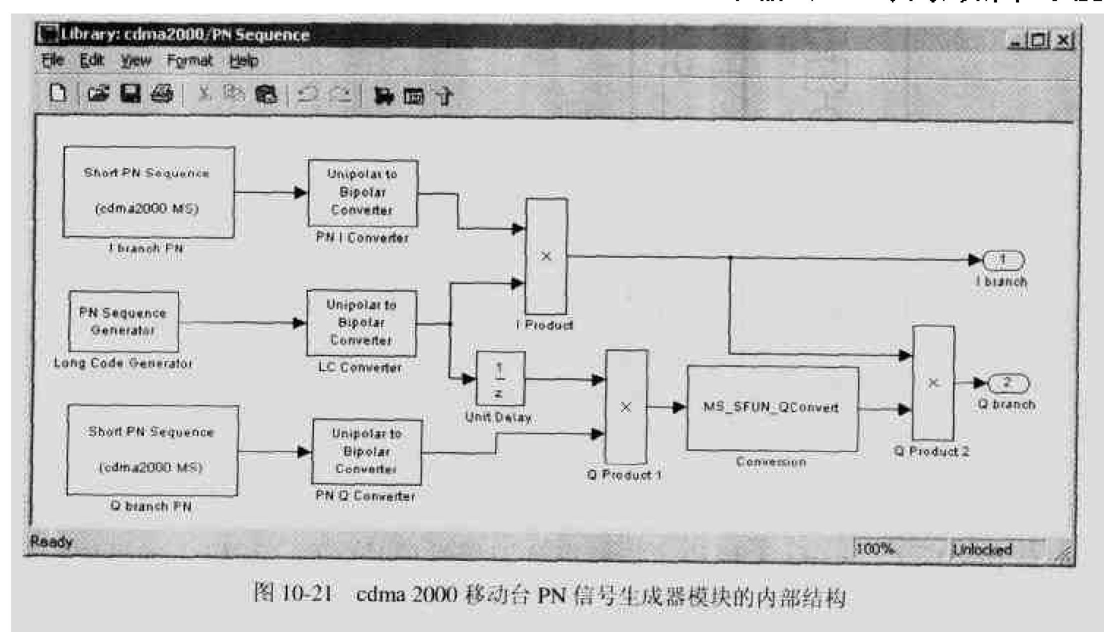



图 10-21　cdma 2000 移动台 PN 信号生成器模块的内部结构

图 10-22 所示是 cdma 2000 协议规范制定的长码产生器的结构。从图中可以看到，cdma 2000 使用的长码是由一个 42 位移位寄存器和一个 42 位掩码计算之后得到的序列。这个 42 位移位寄存器使用的生成多项式是：

$$p(x) = x^{42} + x^{35} + x^{33} + x^{31} + x^{27} + x^{26} + x^{25} + x^{22} + x^{21} + x^{19} + x^{18}$$

$$+ x^{17} + x^{16} + x^{10} + x^{7} + x^{6} + x^{5} + x^{3} + x^{2} + x + 1$$

另外，长码产生器中的 42bit 掩码可以是公共掩码，也可以是私有掩码，前者通过移动台的设备序列号（ESN，Equipment Serial Number）计算得到，后者则通过加密过程获得，并且这个掩码的前面两位等于 01。在这里我们采用私有掩码方式，其中私有掩码的后面 40bit 通过一个随机过程来模拟产生。

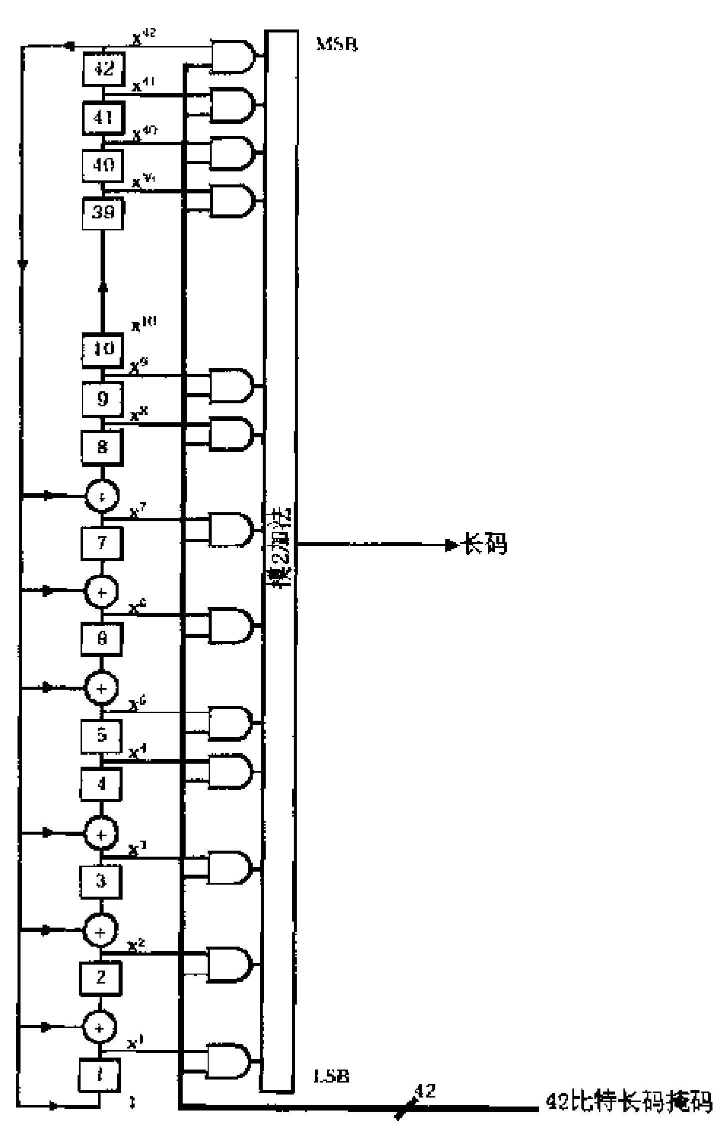



图 10-22　cdma 2000 长码产生器

长码生成模块的输出信号是一个单极性信号序列，因此还需要通过一个极性转换器（Unipolar to Bipolar Converter）模块才能够得到双极性信号序列。长码生成模块和极性转换模块的参数设置分别如表 10-14 和表 10-15 所示。

**表 10-14　　长码生成器（Long Code Generator）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | PN Sequence Generator |
| Generator polynomial | xLongCodeGenerator |
| Initial states | [zeros(1,41) 1] |
| Shift (or mask) | xLongCodeMask |
| Sample time | xSampleTime |
| Frame-based outputs | Unchecked |

**表 10-15　　极性转换器（Unipolar to Bipolar Converter）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Unipolar to Bipolar Converter |
| M-ary number | 2 |
| Polarity | Negative |

长码序列与 I 支路 PN 序列生成模块产生的短码序列相乘就得到了 cdma 2000 移动台 PN 信号生成器模块 I 支路的输出信号，而 Q 支路的输出信号还需要实施一些更加复杂的变换。I 支路 PN 序列生成模块和 Q 支路 PN 序列生成模块是我们自己设计的短 PN 序列生成（Short PN Sequence）模块，它能够根据生成多项式产生长度为 $2^{15}$ 的输出信号序列。关于短 PN 序列生成模块将在后面介绍，这个模块只有一个参数，表 10-16 和表 10-17 分别是 I 支路 PN 序列生成模块和 Q 支路 PN 序列生成模块的参数设置情况。

**表 10-16　　I 支路 PN 序列生成（I branch PN）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Short PN Sequence |
| Generator Polynomial | xPNGeneratorI |

**表 10-17　　Q 支路 PN 序列生成（Q branch PN）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Short PN Sequence |
| Generator Polynomial | xPNGeneratorQ |

cdma 2000 移动台 PN 信号生成器模块 Q 支路的输出信号比较复杂，它通过单位时延模块（Unit Delay）把长码生成模块的输出信号延时一个码片后与 Q 支路 PN 序列生成模块的短码序列相乘，然后通过一个 Q 支路信号变换（Conversion）模块，产生的输出信号再与 cdma 2000 移动台 PN 信号生成器模块 I 支路的输出信号相乘，从而得到 Q 支路的输出信号。表 10-18 和表 10-19 分别列出了单位时延模块和 Q 支路信号变换模块的参数设置。

**表 10-18　　单位时延（Unit Delay）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Unit Delay |
| Initial conditions | 0 |
| Sample time (-1 for inherited) | xSampleTime |

表 10-19　　Q 支路信号变换（Conversion）模块的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | S-Function |
| S-function name | MS_SFUN_QConvert |
| S-function parameters | |

Q 支路信号变换模块通过一个名为 MS_SFUN_QConvert 的 M 文件 S-函数对 Q 支路信号进行变换。根据 cdma 2000 协议要求，输入信号首先以 1/2 的比例进行抽取，这个抽取信号再乘于一个长度为 2、下标为 1 的 Walsh 序列（即正负交替的序列），其效果相当于在某个时刻直接把输入信号当作输出信号，而在下一时刻则输出前一时刻输出信号的相反数。M 文件 S-函数 MS_SFUN_Qconvert.m 的代码如下：

```
function [sys,x0,str,ts] = MS_SFUN_QConvert(t,x,u,flag)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% M 文件 S-函数的主体部分
% 函数名称：MS_SFUN_QConvert
% 主要功能：根据输入参数 flag 的数值调用相应的函数
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch flag,
  case 0,
    % 当 flag 等于 0 时调用 mdlInitializeSizes 函数执行初始化
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 2,
    % 当 flag 等于 2 时调用 mdlUpdate 函数计算离散状态的数值
    sys=mdlUpdate(t,x,u);
  case 3,
    % 当 flag 等于 3 时调用 mdlOutputs 函数计算输出信号的数值
    sys=mdlOutputs(t,x,u);
  case {1, 4, 9}
    % 当 flag 等于 1、4 或 9 时没有相应的操作（没有连续状态）
    sys=[];
  otherwise
    % 当 flag 等于其他数值时表示仿真过程出错
    error(['Unhandled flag = ',num2str(flag)]);
end
% end MS_SFUN_QConvert
%
%=============================================================================
% mdlInitializeSizes
% S-函数的初始化
% 向 Simulink 返回 S-函数各种信号的长度、初始设置和抽样时间设置
%=============================================================================
```

```
function [sys,x0,str,ts]=mdlInitializeSizes
% 调用 simsizes 获得一个用于存放长度信息的结构
sizes = simsizes;
% 设置连续状态的个数
sizes.NumContStates = 0;
% 设置 2 个离散状态
sizes.NumDiscStates = 2;
% 设置输出信号的个数为 1
sizes.NumOutputs = 1;
% 设置输入信号的个数为 1
sizes.NumInputs = 1;
% 设置直接反馈的状态：
%    0 表示没有直接反馈
%    1 表示存在直接反馈
sizes.DirFeedthrough = 1;
% 设置抽样时间的个数（大于等于 1）
sizes.NumSampleTimes = 1;
% 通过 simsizes 把 sizes 结构返回给 sys
sys = simsizes(sizes);
%
% 设置 S-函数的初始状态 x0
x0 = [0 0]';
%
% 设置 S-函数的保留参数 str（应该设置为空向量[]）
str = [];
%
% 初始化抽样时间
ts   = [-1 0];
% end mdlInitializeSizes
%
%=============================================================================
% mdlUpdate
% 更新 S-函数的离散状态并且向 Simulink 返回这些状态的数值
%=============================================================================
function sys=mdlUpdate(t,x,u)
% 计算 S-函数的离散状态并且通过 sys 参数返回给 Simulink
if x(1) == 0
    x(2) = -1*u;
    x(1) = 1;
else
```

```
    x(1) = 0;
end
sys = x;
% end mdlUpdate
%
%==============================================================================
% mdlOutputs
% 计算 S-函数的输出信号并且返回给 Simulink 作为模块的输出
%==============================================================================
function sys=mdlOutputs(t,x,u)
% 计算 S-函数的输出信号并且通过 sys 参数返回给 Simulink
if x(1) == 0
    sys = u;
else
    sys = x(2);
end
% end mdlOutputs
```

在 MS_SFUN_Qconvert.m 中，设置了两个离散状态，其中第一个状态表示是否输出当前输入信号：当它等于 0 时输出信号等于输入信号，当它等于 1 时输出信号等于第二个状态中保存的数据。由于第一个状态再每得到一个输入信号时翻转一次，因此 S-函数的输出信号交替等于输入信号。同时，当第一个状态等于 0 时，第二个状态就保存了当前输入信号的相反数，这个数据就是下一个输出信号的数值。

下面我们来看看短 PN 序列（Short PN Sequence）生成模块，其模块框图和参数设置对话框如图 10-23 所示。

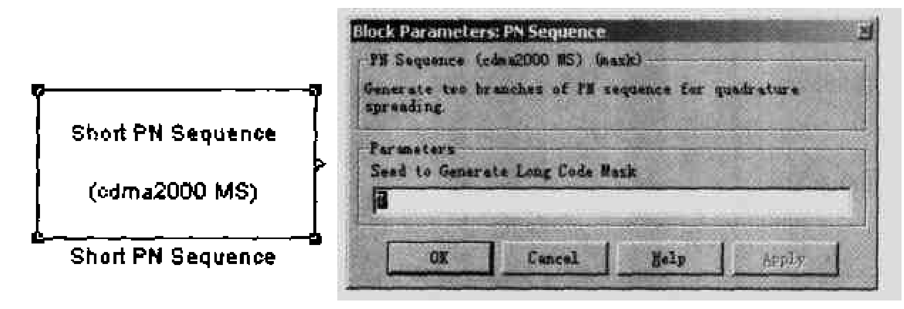



图 10-23　cdma 2000 基站短 PN 序列生成模块及其参数设置对话框

图 10-24 所示是 cdma 2000 基站短 PN 序列生成模块的内部结构。在 cdma 2000 基站短 PN 序列生成模块中，PN 序列（PN Sequence）生成器产生一个长度为 $2^{15}-1$ 的 PN 序列帧，然后通过零插入模块（Insert Zero）在其中插入一个 0，使得帧长度变成 $2^{15}$，同时每个序列的头部等于一个 1 后面跟着 15 个零。最后，短 PN 序列生成模块通过缓冲去除（Unbuffer）模块把帧格式的 PN 序列转换成抽样序列。

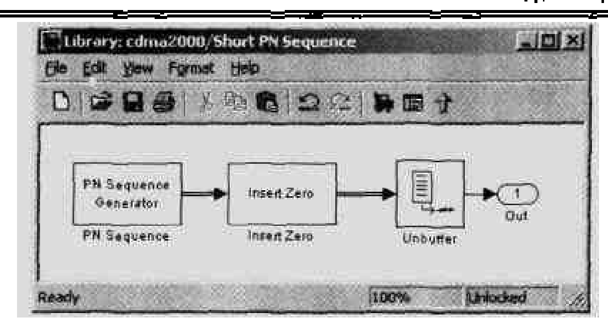



图 10-24 cdma 2000 基站短 PN 序列生成模块的内部结构

为了使短 PN 序列生成模块输出序列的码片速率等于 1.2288Mchip/s，一个长度为 $2^{15}$ 的数据帧的周期应该等于 $2^{15}/1228800$。由于 PN 序列生成器的输出信号是长度为 $2^{15}-1$ 的数据帧，因此 PN 序列生成器的抽样时间就应该是 $2^{15}/1228800/(2^{15}-1)$。表 10-20 和表 10-21 分别列出了 PN 序列生成器和零插入模块的参数设置。

表 10-20 PN 序列生成器（PN Sequence）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | PN Sequence Generator |
| Generator polynomial | xGenerator |
| Initial states | [zeros(1,14) 1] |
| Shift (or mask) | [zeros(1,14) 1] |
| Sample time | 2^15/1228800/(2^15-1) |
| Frame-based outputs | Checked |
| Samples per frame | 2^15-1 |
| Reset on nonzero input | Unchecked |

表 10-21 零插入模块（Insert Zero）的参数设置

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Insert Zero |
| Insert zero vector | [1 0 ones(1,2^15-2)]' |

cdma 2000 基站短 PN 序列生成模块封装成子系统之后只有一个参数 Generator Polynomial，与之对应的子系统内部变量等于 xGenerator。这个封装子系统没有初始化代码，它通过 disp('Short PN Sequence\n\n(cdma 2000 MS)') 绘制图标。

最后，我们把整个 PN 信号生成器转化成一个封装子系统。我们把 Icon 面板的 Drawing commands 设置为 disp('PN Sequence\n\n(cdma 2000 MS)')，Initialization 面板的 Initialization commands 设置为 MS_PNSequence_Initialization，并且创建一个封装子系统参数 Seed to Generate Long Code Mask，与之相对应的内部变量是 xSeed。下面的程序段是 MS_PNSequence_Initialization.m 文件的代码，它用于设置长码和短码的生成多项式。

```
%===========================================================
% MS_PNSequence_Initialization.m
%===========================================================
% 长码产生器的生成多项式
% p(x) = x42 + x35 + x33 + x31 + x27 + x26 + x25 + x22 + x21 + x19 + x18 + x17 + x16 + x10 + x7 + x6 +
```

```
x5 + x3 + x2 + x1 + 1
    xLongCodeGenerator = [42 35 33 31 27 26 25 22 21 19 18 17 16 10 7 6 5 3 2 1 0];
    % 长码掩码，它根据随机数种子产生一个随机序列
    xLongCodeMask = [0 1 randint(1,40,2,xSeed)];

    % I 支路短码的生成多项式
    % PI(x) = x15 + x13 + x9 + x8 + x7 + x5 + 1
    xPNGeneratorI = [15 13 9 8 7 5 0];
    % 支路短码的生成多项式
    % PQ(x) = x15 + x12 + x11+ x10 + x6 + x5 + x4 + x3 + 1
    xPNGeneratorQ = [15 12 11 10 6 5 4 3 0];

    % 码片周期
    xSampleTime = 1/1228800;
```

现在我们完成了 cdma 2000 基站 PN 信号生成器的设计。这个 PN 信号生成器产生的两路 PN 序列不仅可以用于在发送端对信号进行扰码，还可以应用在接收端，用于对接收到的信号实施解扰。

## 10.2.7 信号调制模块

cdma 2000 基站调制模块采用 PN 信号生成器的 I 支路和 Q 支路 PN 序列对反向信道数据进行扰码，然后把信号调制到载波频率 $f_c$ 上形成射频输出信号。cdma 2000 基站调制模块及其参数设置对话框如图 10-25 所示。

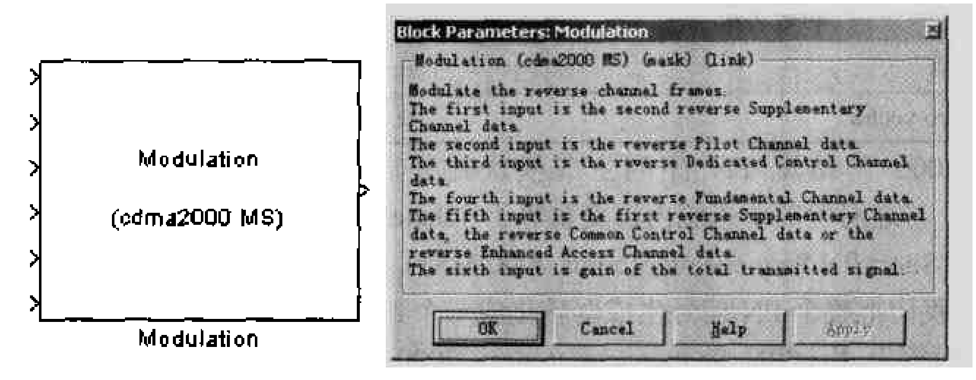



图 10-25 cdma 2000 基站调制模块及其参数设置对话框

在 cdma 2000 基站调制模块中采用基带信号来表示调制后的射频信号，以节约仿真时间。cdma 2000 基站调制模块的内部结构如图 10-26 所示。从图中可以看到，cdma 2000 基站调制模块共有 6 个输入端口，其中前 5 个端口分别表示反向补充信道 2（SCH2）、反向导频信道（PCH）、反向专用控制信道（DCCH）、反向基本信道（FCH）以及反向补充信道 1/公共控制信道/增强接入信道（DCH1/CCCH/EACH）的输入信号。

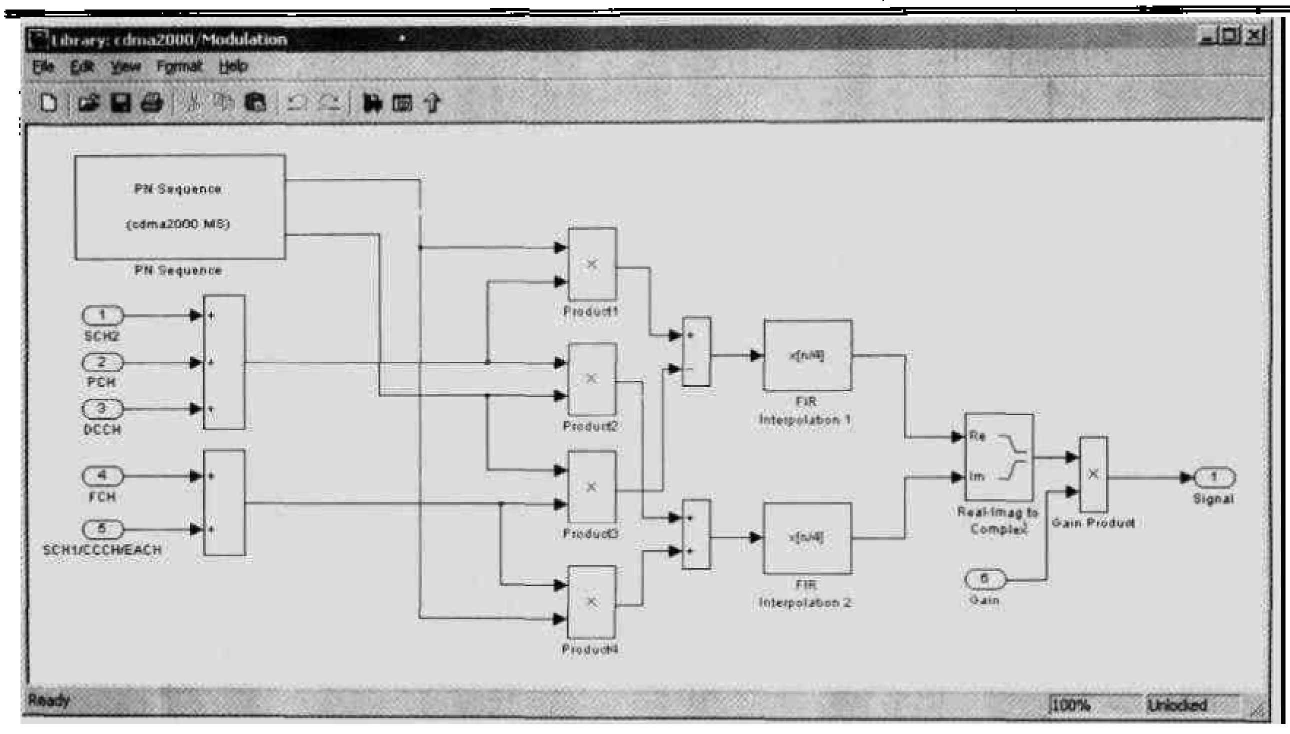



图 10-26　cdma 2000 基站调制模块的内部结构

在 cdma 2000 基站调制模块中，前 3 个输入信号之和成为 I 支路调制数据，后两个输入信号之和则成为 Q 支路调制数据，这两个支路的信号分别与 PN 信号生成器的 I 支路和 Q 支路 PN 序列执行乘法和加法运算，得到两个支路的信号序列。这两个信号序列分别通过 FIR 滤波器（FIR Interpolation 1 和 FIR Interpolation 2）进行滤波，然后把它们组合成一个复信号，乘以一个发射增益之后就形成了基带调制信号。表 10-22 列出了 FIR 滤波器模块的参数设置情况，其中 xFIR 表示滤波器的系数。

**表 10-22　　FIR 滤波器（FIR Interpolation）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | FIR Interpolation |
| FIR filter coefficients | xFIR |
| Interpolation factor | 4 |
| Framing | Maintain input frame size |
| Output buffer initial conditions | 0 |

cdma 2000 协议对基带滤波器的频率响应特性进行了严格的限制，如图 10-27 所示。根据这个协议的规范，基带滤波器是一个低通滤波器，它的通频带范围是 $0 \leqslant f \leqslant f_p$，其中 $f_p = 590\text{kHz}$。在这个通频带范围内，滤波器的归一化频率响应局限在 $\pm\delta_1$，$\delta_1$=1.5dB。另外，滤波器的抑制频带范围是 $f \geqslant f_s$，$f_s = 740\text{kHz}$。基带滤波器在抑制频带内的频率衰减幅度应该达到 $\delta_2 = 40\text{dB}$ 以上。

最后我们把这个子系统转化成一个封装子系统。打开封转子系统的 Edit mask 窗口，把 Icon 面板的 Drawing commands 设置为 disp('Modulation\n\n(cdma 2000 MS)')，并且把 Initialization 面板的 Initialization commands 设置为 MS_Modulation_Initialization。这个子系统不需要设置额外的参数。

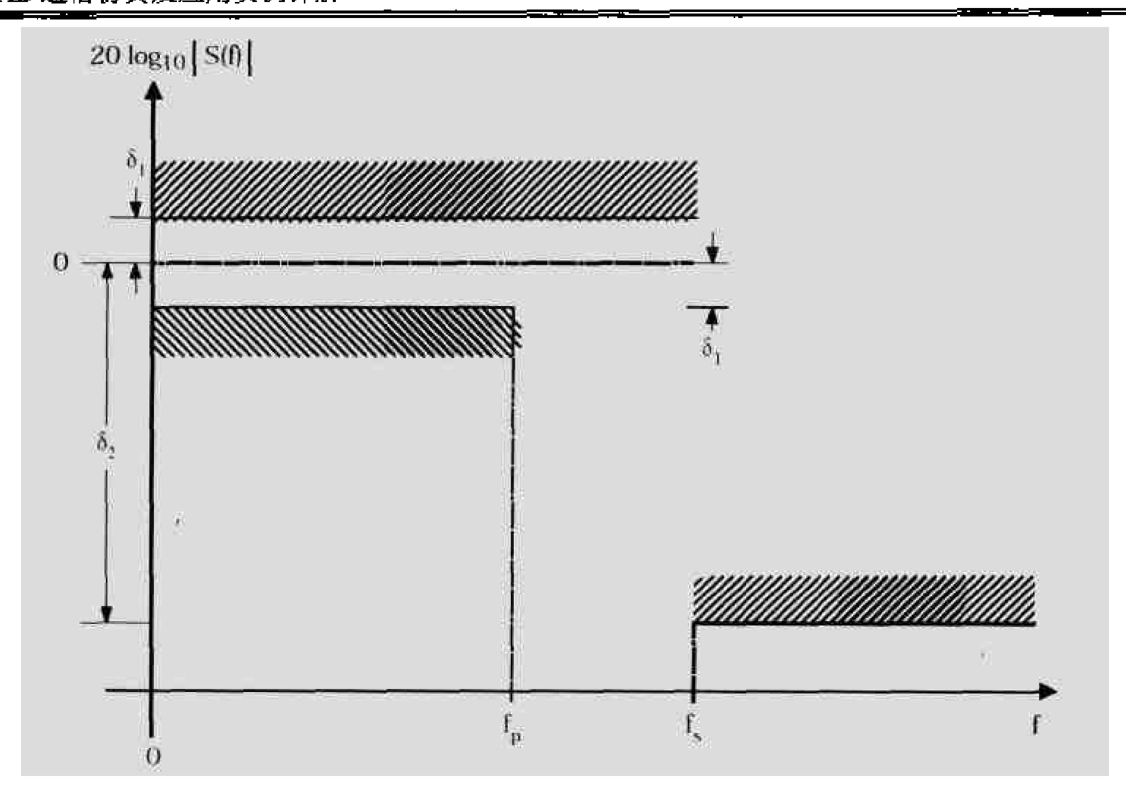



图 10-27 基带滤波器频率响应特性

M 文件 MS_Modulation_Initialization.m 用于设置滤波器的系数 xFIR，其代码如下：

```
%=====================================================================
% MS_Modulation_Initialization.m
%=====================================================================
% 设置 FIR 滤波器的系数
xFIR = [-0.025288315, -0.034167931, -0.035752323, -0.016733702, 0.021602514, 0.064938487, 0.091002137,
0.081894974, 0.037071157, -0.021998074, -0.060716277, -0.051178658, 0.007874526, 0.084368728, 0.126869306,
0.094528345, -0.012839661, -0.143477028, -0.211829088, -0.140513128, 0.094601918, 0.441387140,
0.785875640, 1.0, 1.0, 0.785875640, 0.441387140, 0.094601918, -0.140513128, -0.211829088, -0.143477028,
-0.012839661, 0.094528345, 0.126869306, 0.084368728, 0.007874526, -0.051178658, -0.060716277,
-0.021998074, 0.037071157, 0.081894974, 0.091002137, 0.064938487, 0.021602514, -0.016733702, -0.035752323,
-0.034167931, -0.025288315];
```

cdma 2000 协议对滤波器使用的参数 xFIR 进行了定义，它的数值就是我们在 M 文件 MS_Modulation_Initialization.m 中定义的数值。

## 10.2.8 初始化模块

在前面创建的各个模块都有一些相同的参数，如无线配置方式、数据帧长度等。当这些模块处于同一个仿真模型中时，它们的参数设置应该相同。如果更改了其中某个模块的某一个参数，其他模块相应参数的数值也得跟着发生改变。为此，我们设计一个初始化模块，对整个仿真模型中相同的参数提供一个统一的平台。图 10-28 所示是 cdma 2000 基站初始化模块及其参数设置对话框。

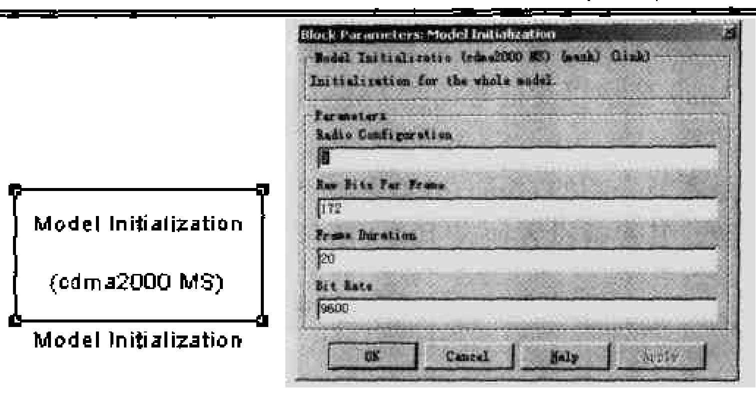



图 10-28 cdma 2000 基站初始化模块及其参数设置对话框

cdma 2000 基站初始化模块是一个空的子系统，里面没有任何的模块，也没有输入输出端口。为了实现上述功能，我们把这个空系统转换成一个封装子系统，并且把封装子系统 Icon 面板中的 Drawing commands 设置为 disp('Model Initialization\n\n(cdma 2000 MS)')，并且创建 4 个封装参数 Radio Configuration、Raw Bits Per Frame、Frame Duration 和 Bit Rate，与之相应的内部变量参数是 xRC、xFrameDuration、xBitRate 和 xFrameLength。同时，我们把 Initialization 面板的 Initialization commands 设置为如下代码：

```
assignin('base', 'mRC', xRC);
assignin('base', 'mFrameDuration', xFrameDuration');
assignin('base', 'mBitRate', xBitRate');
assignin('base', 'mFrameLength', xFrameLength);
```

上述命令使得仿真模型在初始化过程中自动创建 4 个工作区变量（mRC、mFrameDuration、mBitRate 和 mFrameLength），这些变量的数值等于初始化模块的参数设置。这样一来，我们就可以在仿真模型的各个模块中使用这些工作区变量。当需要修改参数设置时，只需在初始化模块中更改相应变量的数值，这个更改将通过初始化命令传递到相应的工作区变量中，进而反映到仿真模型的各个子模块，从而大大简化了参数设置过程。

下面通过一个示例程序来说明前面介绍的各个模块的使用方法。在这个实例程序中，我们将实现一个 cdma 2000 基站发射机，它将模拟反向基本信道射频信号产生的全过程。cdma 2000 基站发射机的模块框图如图 10-29 所示。

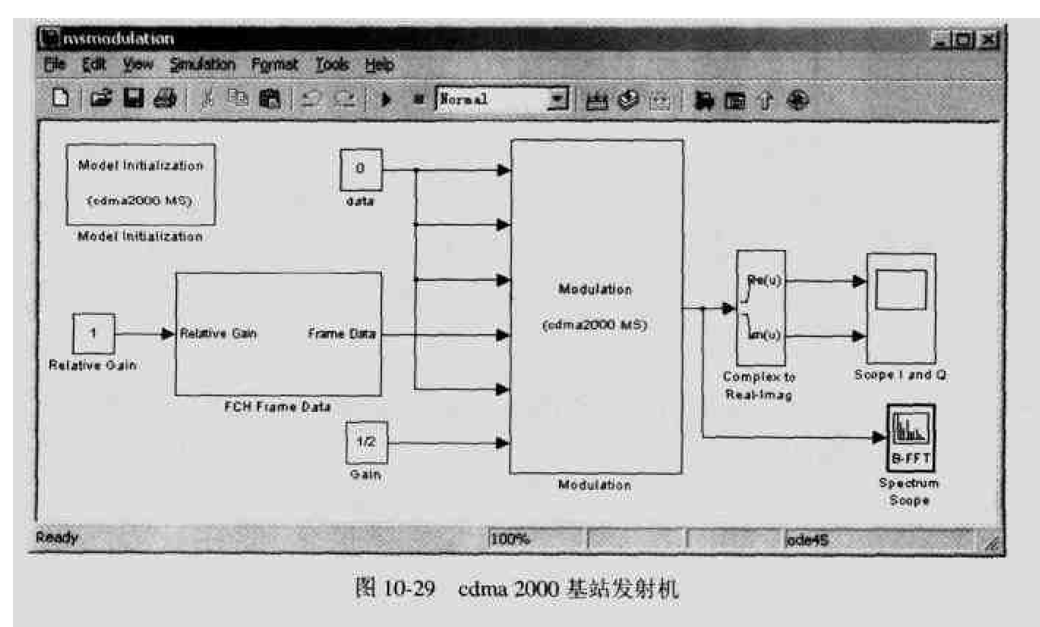



图 10-29 cdma 2000 基站发射机

cdma 2000 基站发射机由 3 部分组成，包括初始化（Model Initialization）模块、数据帧处理（FCH Frame Data）模块以及信号调制（Modulation）模块。最后可以通过示波器（Scope I and Q）和频谱示波器（Spectrum Scope）观察调制信号的波形和频谱。

初始化模块就是本节介绍的 cdma 2000 基站初始化模块，通过它可以方便地设置仿真模型中的一些常用参数，其参数设置如表 10-23 所示。

**表 10-23　　初始化（Model Initialization）模块的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Model Initialization |
| Radio Configuration | 3 |
| Raw Bits Per Frame | 172 |
| Frame Duration | 20 |
| Bit Rate | 9600 |

数据帧处理模块实现对反向基本信道数据帧的 CRC 编码、卷积编码、信号交织以及正交扩频功能，最后输出传输速率为 1.2288Mchip/s 的数据流。在数据帧处理模块中，我们使用了前面设计的 cdma 2000 基站 CRC 编码器（CRC）模块、卷积编码器（Encoder）模块、信号交织（Interleaver）模块以及正交扩频（Orthogonal Spreading）模块。这些模块具有相同的参数，我们把 Radio Configuration 设置为 mRC，Bit Rate 设置为 mBitRate，Raw Bits Per Frame 设置为 mFrameLength，Frame Duration 设置为 mFrameDuration。这样，当我们需要更改其中某个参数的数值时，只需要在初始化模块中更改相应参数的数值，而这种更改将通过工作区变量传递到这些模块中，从而免除了对每个模块的参数更改操作。

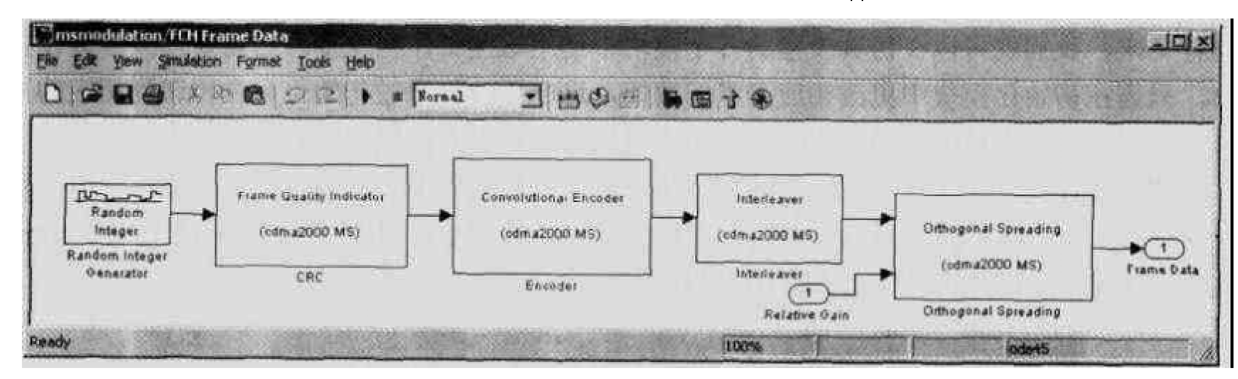



图 10-30　数据帧处理模块

另外，对于正交扩频模块，它还有一个参数 Channel Type，它表示信道的类型，在这里把它设置为 Reverse Fundamental Channel，表示输入信号是反向基本信道的数据帧。

在数据帧处理模块中，通过随机整数生成器（Random Integer Generator）产生一个长度为 mFrameLength 的数据帧，这个数据帧的周期等于 mFrameDuration 毫秒。表 10-24 列出了随机整数生成器的参数设置情况。

**表 10-24　　随机整数生成器（Random Integer Generator）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Random Integer Generator |
| M-ary number | 2 |

续表

| 参数名称 | 参数值 |
|---|---|
| Initial seed | 37 |
| Sample time | mFrameDuration/1000/mFrameLength |
| Frame-based outputs | Checked |
| Samples per frame | mFrameLength |

信号调制模块直接使用了前面介绍的 cdma 2000 基站信号调制模块。由于这个仿真模型只模拟反向基本信道，因此 cdma 2000 基站信号调制模块的第 1、2、3、5 号端口的输入信号是 0，第 4 个端口的输入信号是数据帧处理模块的输出信号，而第 6 个端口则输入常数 1 表示发射信号的总增益。

最后，信号调制模块的输出信号分别进入示波器模块和频谱示波器模块。把调制信号的输出信号通过一个转换模块（Complex to Real-Imag），分别得到调制信号的实部和虚部，分别在示波器的两个坐标中显示出来。图 10-31 所示是示波器模块的输出波形。

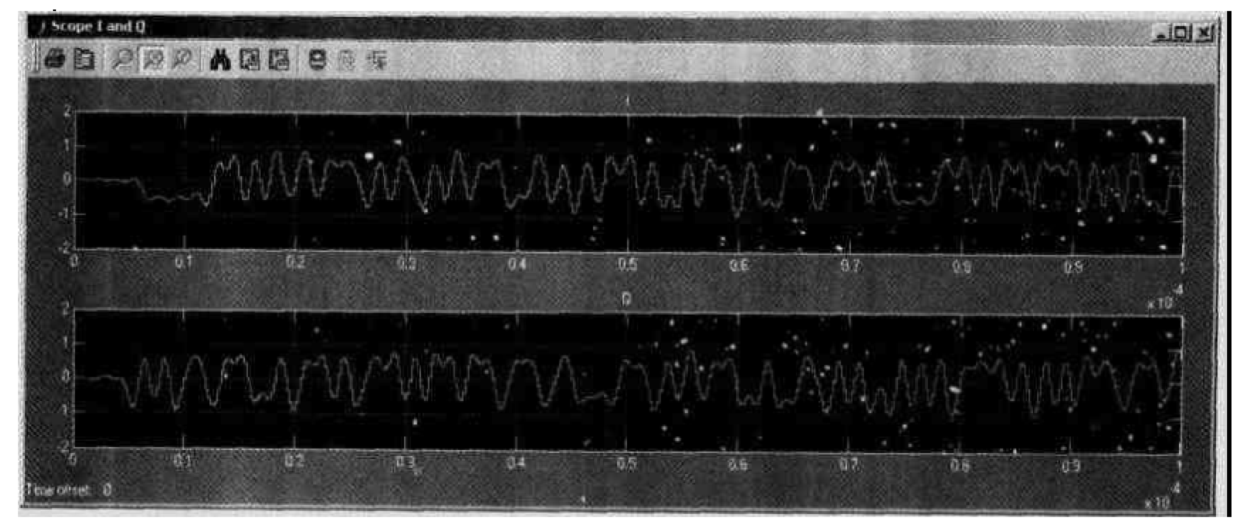


图 10-31 示波器的输出信号

从图 10-31 所示中可以看到，信号调制模块产生的基带调制信号在经过滤波之后变得比较平缓，同时这两个支路的信号是不相同的，这一方面是由于采用了不同的扰码信号，另一方面也因为它们承载了不同信道的数据帧。

为了观察基带调制信号的频谱状况，我们还把信号调制模块的输出信号连接到一个频谱示波器。频谱示波器能够对信号实施频谱分析，然后把分析结果显示出来。频谱示波器（Spectrum Scope）的参数设置如表 10-25 所示。

**表 10-25 频谱示波器（Spectrum Scope）的参数设置**

| 参数名称 | 参数值 |
|---|---|
| 模块类型 | Spectrum Scope |
| Frequency units | Hertz |
| Frequency range | [-Fs/2...Fs/2] |
| Inherit sample increment from input | Checked |
| Amplitude scaling | dB |

续表

| 参数名称 | 参数值 |
| --- | --- |
| Minimum Y-limit | -65 |
| Maximum Y-limit | 15 |
| Y-axis title | Magnitude, dB |

图 10-32 所示是频谱示波器的输出信号波形。从图 10-31 所示中可以看到，这个 cdma 2000 基站产生的基带调制信号的频谱与协议规范的要求是一致的，它在抑制频谱范围内的衰减幅度在 40dB 以上。

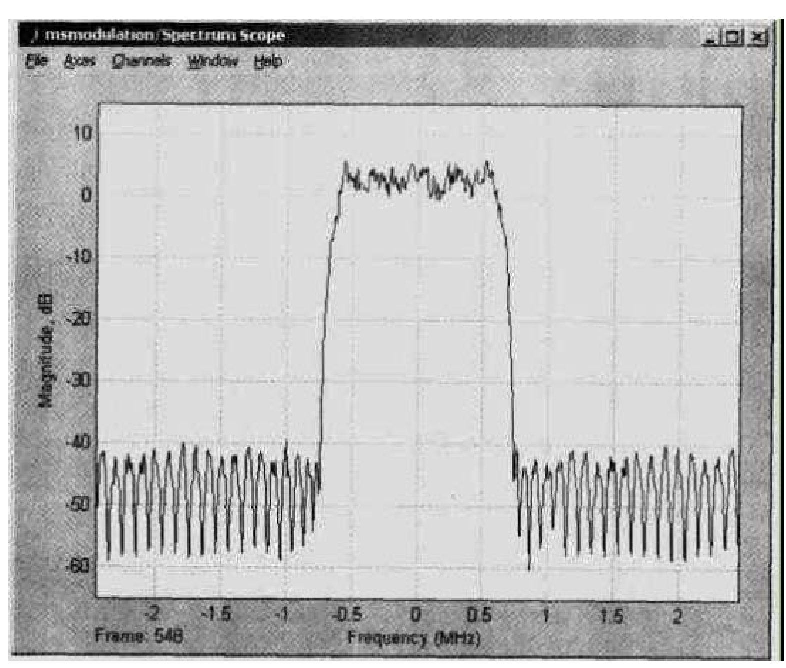



图 10-32　频谱示波器的输出波形

## 10.2.9　基站接收器

基站接收器接收反向信道的射频信号，通过解调获得编码符号，然后再通过解交织、卷积译码和 CRC 译码等过程还原得到原始的数据帧。这个过程基本上就是移动台调制器的逆过程。

在前面已经提到过，假设 $d_I(t)$ 和 $d_Q(t)$ 是反向业务信道扰码之前两个支路的输出信号，$p_I(t)$ 和 $p_Q(t)$ 是用于扰码的两个短码序列，由此得到两个信号序列 $s_I(t)$ 和 $s_Q(t)$，其中：

$$\begin{cases} s_I(t) = p_I(t)d_I(t) - p_Q(t)d_Q(t) \\ s_Q(t) = p_Q(t)d_I(t) + p_I(t)d_Q(t) \end{cases}$$

这两个信号序列在基带滤波之后调制到载波频率 $f_c$ 上，形成射频输出信号 $s(t) = s_I(t)\cdot\cos(2\pi f_c t) + s_Q(t)\cdot\sin(2\pi f_c t)$。

在忽略噪声和干扰的情况下，基站接收器接收到的信号 $r(t)$ 等于 $s(t)$。对 $r(t)$ 实施相干解调或非相干解调之后就得到了两个支路的信号序列 $s_I(t)$ 和 $s_Q(t)$。对于基带仿真，这个解调过程可以忽略，从而直接得到这两个信号序列。

对于 $s_I(t)$ 和 $s_Q(t)$，我们需要从中得到反向业务信道扰码前的编码符号序列 $d_I(t)$ 和 $d_Q(t)$。为此，把 $s_I(t)$ 和 $s_Q(t)$ 分别乘于扰码序列 $p_I(t)$ 和 $p_Q(t)$，并且在一个周期 $T$（一般情况下 $s_I(t)$ 和 $s_Q(t)$ 的周期 $T$ 远大于扰码序列的码片周期 $T_c$）内积分得到：

$$\begin{aligned}\int_{kT}^{(k+1)T} s_I(t)\cdot p_I(t)\mathrm{d}t &= \int_{kT}^{(k+1)T} p_I(t)\cdot d_I(t)\cdot p_I(t)\mathrm{d}t - \int_{kT}^{(k+1)T} p_Q(t)\cdot \mathrm{d}_Q(t)\cdot p_I(t)\mathrm{d}t\\ &= T\cdot d_I(t) - \int_{kT}^{(k+1)T} p_Q(t)\cdot d_Q(t)\cdot p_I(t)\mathrm{d}t\end{aligned}$$

$$\begin{aligned}\int_{kT}^{(k+1)T} s_Q(t)\cdot p_Q(t)\mathrm{d}t &= \int_{kT}^{(k+1)T} p_Q(t)\cdot d_I(t)\cdot p_Q(t)\mathrm{d}t + \int_{kT}^{(k+1)T} p_I(t)\cdot d_Q(t)\cdot p_Q(t)\mathrm{d}t\\ &= T\cdot d_I(t) + \int_{kT}^{(k+1)T} p_I(t)\cdot d_Q(t)\cdot p_Q(t)\mathrm{d}t\end{aligned}$$

将上面的两个式子相加之后得到：

$$d_I(t) = \frac{1}{2T}\left(\int_{kT}^{(k+1)T} s_I(t)\cdot p_I(t)\mathrm{d}t + \int_{kT}^{(k+1)T} s_Q(t)\cdot p_Q(t)\mathrm{d}t\right)$$

同理，对于 $d_Q(t)$ 我们有：

$$d_Q(t) = \frac{1}{2T}\left(\int_{kT}^{(k+1)T} s_Q(t)\cdot p_I(t)\mathrm{d}t - \int_{kT}^{(k+1)T} s_I(t)\cdot p_Q(t)\mathrm{d}t\right)$$

从而可以得到编码符号序列 $s_I(t)$ 和 $s_Q(t)$。需要注意的是，在上面的计算过程中我们假定基站接收器能够获得与移动台发射机相同的扰码序列 $p_I(t)$ 和 $p_Q(t)$，这需要在发送端和接收端之间建立一种信号同步机制。

由于 $d_I(t)$ 和 $d_Q(t)$ 中都可能包含了多个信道，因此还需要从中换远处每一个信道的信号。假设 $d_1(t)$ 和 $d_2(t)$ 是两个反向信道的数据流，它们分别采用 Walsh 序列 $W_i^n(t)$ 和 $W_j^n(t)$ 进行扩展，然后复用到 I 支路信号中，即 $d_I(t) = d_1(t)\cdot W_i^n(t) + d_2(t)\cdot W_j^n(t)$。利用 Walsh 码的正交性我们可以从 $d_I(t)$ 中解出 $d_1(t)$ 和 $d_2(t)$，即：

$$\begin{aligned}\int_0^T d_I(t)\cdot W_i^n(t)\mathrm{d}t &= \int_0^T d_1(t)\cdot W_i^n(t)\cdot W_i^n(t)\mathrm{d}t + \int_0^T d_2(t)\cdot W_j^n(t)\cdot W_i^n(t)\mathrm{d}t\,,\\ &= T\cdot d_1(t)\end{aligned}$$

上式第二个积分项等于 0，从而得到 $d_1(t)$ 和 $d_2(t)$。

每个信道的数据 $d_1(t)$ 和 $d_2(t)$ 都是经过编码和交织的信号序列，这时候就可以按照 cdma 2000 协议规范中对移动台发射机要求实施相反的处理过程，最终得到原始的数据帧。例如，对于 RC3 反向基本信道和反向补充信道，信道解调数据首先经过解交织器，得到交织前的信号序列。如果这个反向业务信道帧在调制过程中实施了抽取操作，基站接收器应该按照抽取比例每隔一定符号添加一个 0。此后是去重复过程，它从具有重复信号的数据帧中删除重复的信号。最后是信号的卷积译码和 CRC 译码过程。卷积译码可以采用硬判决方式，也可以采用 Viterbi 算法实施软判决译码。如果采用软判决译码方式，信号解调过程中得到的数

据应该是对 0 和 1 的判决尺度。

在上面的讨论过程中我们都没有考虑噪声和干扰。在具有高斯白噪声和瑞利衰落的信道条件下，基站接收器接收到的是噪声加干扰信号。即使如此，上述方法依然适用，cdma 2000 系统的各种措施都围绕着如何尽可能地降低噪声和干扰带来的影响。

另外，cdma 系统广泛采用了 Rake 接收技术，它能够从多径传输信号中分离出可用信号，并且把这些信号叠加起来，增强了接收信号的强度及其信噪比。

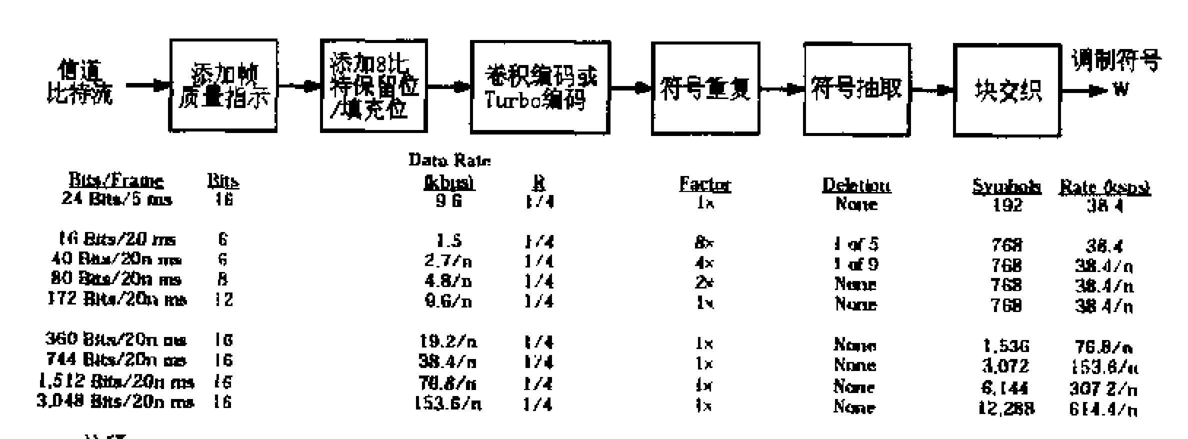
以上我们介绍了基站接收器的工作流程。由于基站接收器的建模和仿真与移动台发射机具有很大的相似性，限于篇幅就不介绍基站接收器的实现过程。

## 10.3 cdma 2000 前向业务信道

cdma 2000 前向信道有 9 种不同的无线配置（RC1~9），其中前两种无线配置（RC1 和 RC2）与 IS-95 兼容，RC3、RC4 和 RC5 适用于单载波的 cdma 2000 1x，后面 4 种无线配置（RC6~9）则适用于三载波方式的 cdma 2000 3x。

跟 cdma 2000 反向信道类似地，cdma 2000 前向信道数据帧也要经过 CRC 编码、卷积编码（或 Turbo 编码）、信号重复、信号抽取、信号交织等过程，不同的信道采用不同的 Walsh 码（或准正交函数）进行扩频，并且通过 PN 码序列进行扰码，在经过基带滤波和载波调制之后就成为无线射频信号。由于 cdma 2000 前向信道与反向信道具有很大的相似性，因此本节我们只介绍 cdma 2000 前向业务信道的基本原理。关于 cdma 2000 前向业务信道仿真模型的设计方法，读者可以参照前面的内容自己完成。

图 10-33 所示是 RC3 前向业务信道数据帧的处理流程。从图 10-33 所示中可以看到，RC3 前向业务信道每个数据帧的长度不等，每帧的周期可以是 5ms、20ms、40ms 或 80ms。但是对于前向基本信道，它的周期只能是 5ms 或 20ms，且每帧数据的长度不超过 172bit。因此，RC3 前向基本信道的最高数据传输速率是 9600bit/s。而对于 RC3 前向补充信道，它的最高传输速率可以达到 153.6kbit/s。




| Bits/Frame | Bits | Data Rate (kbps) | R | Factor | Deletion | Symbols | Rate (ksps) |
|---|---|---|---|---|---|---|---|
| 24 Bits/5 ms | 16 | 9.6 | 1/4 | 1x | None | 192 | 38.4 |
| 16 Bits/20 ms | 6 | 1.5 | 1/4 | 8x | 1 of 5 | 768 | 38.4 |
| 40 Bits/20n ms | 6 | 2.7/n | 1/4 | 4x | 1 of 9 | 768 | 38.4/n |
| 80 Bits/20n ms | 8 | 4.8/n | 1/4 | 2x | None | 768 | 38.4/n |
| 172 Bits/20n ms | 12 | 9.6/n | 1/4 | 1x | None | 768 | 38.4/n |
| 360 Bits/20n ms | 16 | 19.2/n | 1/4 | 1x | None | 1,536 | 76.8/n |
| 744 Bits/20n ms | 16 | 38.4/n | 1/4 | 1x | None | 3,072 | 153.6/n |
| 1,512 Bits/20n ms | 16 | 76.8/n | 1/4 | 1x | None | 6,144 | 307.2/n |
| 3,048 Bits/20n ms | 16 | 153.6/n | 1/4 | 1x | None | 12,288 | 614.4/n |

注释：

1、n是帧的持续时间（表示为20毫秒的整数倍）。对于40比特的帧，n = 1或2；对于长度大于40比特的帧，n = 1，2或4。

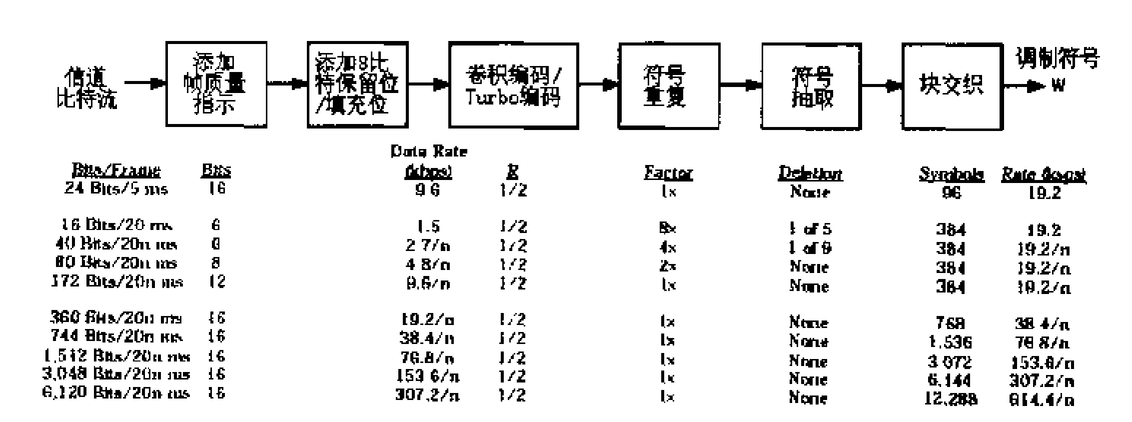
2、5毫秒帧只适用于前向基本信道；另外，前向基本信道只能使用n = 1时16至172比特长度的帧。

3、大于360比特的前向补充信道可以采用Turbo编码方式，除此之外只能采用卷积编码。

4、卷积编码需要8比特的填充位；Turbo编码8比特填充位的前两个比特是保留位，其余的6比特是Turbo编码器产生的数据。

图 10-33　RC3 前向基本信道和前向补充信道

RC4 前向业务信道可以接受的数据帧长度与 RC3 相同（RC4 的数据帧长度还可以等于 6120bit），如图 10-34 所示。由于 RC4 采用了 1/2 码率的卷积编码，因此其最大传输速率（前向补充信道）可以达到 307.2kbit/s，这个速度是 RC3 的两倍。但是对于前向基本信道，它的最大传输速率仍然只有 9600bit/s。

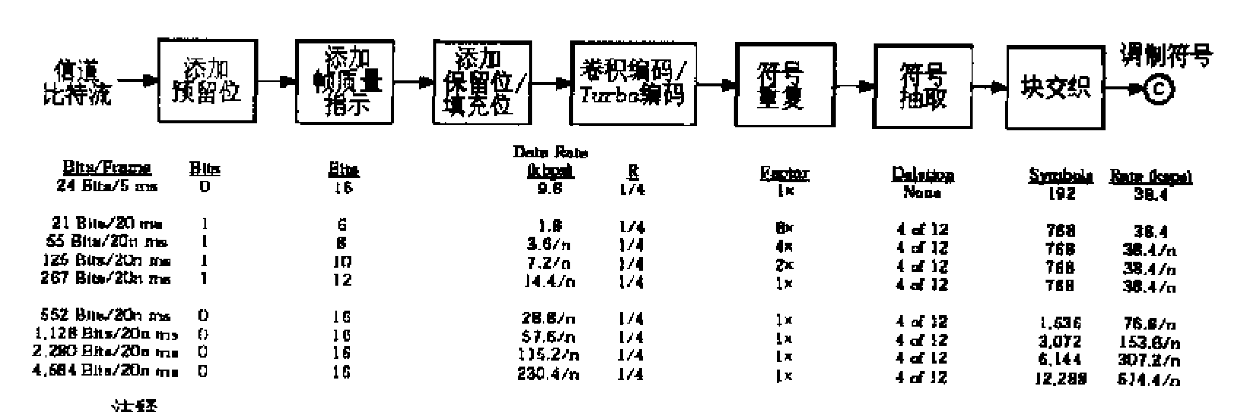



| Bits/Frame | Bits | Data Rate (kbps) | R | Factor | Deletion | Symbols | Rate (ksps) |
|---|---|---|---|---|---|---|---|
| 24 Bits/5 ms | 16 | 9.6 | 1/2 | 1x | None | 96 | 19.2 |
| 16 Bits/20 ms | 6 | 1.5 | 1/2 | 8x | 1 of 5 | 384 | 19.2 |
| 40 Bits/20n ms | 6 | 2.7/n | 1/2 | 4x | 1 of 9 | 384 | 19.2/n |
| 80 Bits/20n ms | 8 | 4.8/n | 1/2 | 2x | None | 384 | 19.2/n |
| 172 Bits/20n ms | 12 | 9.6/n | 1/2 | 1x | None | 384 | 19.2/n |
| 360 Bits/20n ms | 16 | 19.2/n | 1/2 | 1x | None | 768 | 38.4/n |
| 744 Bits/20n ms | 16 | 38.4/n | 1/2 | 1x | None | 1,536 | 76.8/n |
| 1,512 Bits/20n ms | 16 | 76.8/n | 1/2 | 1x | None | 3,072 | 153.6/n |
| 3,048 Bits/20n ms | 16 | 153.6/n | 1/2 | 1x | None | 6,144 | 307.2/n |
| 6,120 Bits/20n ms | 16 | 307.2/n | 1/2 | 1x | None | 12,288 | 614.4/n |

注释：
1、n是帧的持续时间（表示为20毫秒的整数倍）。对于40比特的帧，n = 1或2；对于长度大于40比特的帧，n = 1，2或4。
2、5毫秒帧只适用于前向基本信道，另外，前向基本信道只能使用n = 1时16至172比特长度的帧。
3、大于360比特的前向补充信道可以采用Turbo编码方式，除此之外只能采用卷积编码。
4、卷积编码需要8比特的填充位，Turbo编码8比特填充位的前两个比特是保留位，其余的6比特是Turbo编码器产生的数据。

图 10-34　RC4 前向基本信道和前向补充信道

RC5 前向业务信道也采用 1/4 码率的卷积编码器。与 RC3 不同的是，RC4 提供了另外一种数据传输速率，前向基本信道的最高传输速率达到了 14400bit/s，同时前向补充信道的最高传输速率也提高到 230.4kbit/s。同样 RC5 前向基本信道数据帧的周期只能等于 5ms 或 20ms，而前向补充信道的周期可以达到 80ms，从而有利于实现数据业务的传输。图 10-35 所示是 RC5 前向业务信道的编码和交织流程。




| Bits/Frame | Bits | Bits | Data Rate (kbps) | R | Factor | Deletion | Symbols | Rate (ksps) |
|---|---|---|---|---|---|---|---|---|
| 24 Bits/5 ms | 0 | 16 | 9.6 | 1/4 | 1x | None | 192 | 38.4 |
| 21 Bits/20 ms | 1 | 6 | 1.8 | 1/4 | 8x | 4 of 12 | 768 | 38.4 |
| 55 Bits/20n ms | 1 | 8 | 3.6/n | 1/4 | 4x | 4 of 12 | 768 | 38.4/n |
| 125 Bits/20n ms | 1 | 10 | 7.2/n | 1/4 | 2x | 4 of 12 | 768 | 38.4/n |
| 267 Bits/20n ms | 1 | 12 | 14.4/n | 1/4 | 1x | 4 of 12 | 768 | 38.4/n |
| 552 Bits/20n ms | 0 | 16 | 28.8/n | 1/4 | 1x | 4 of 12 | 1,536 | 76.8/n |
| 1,128 Bits/20n ms | 0 | 16 | 57.6/n | 1/4 | 1x | 4 of 12 | 3,072 | 153.6/n |
| 2,280 Bits/20n ms | 0 | 16 | 115.2/n | 1/4 | 1x | 4 of 12 | 6,144 | 307.2/n |
| 4,584 Bits/20n ms | 0 | 16 | 230.4/n | 1/4 | 1x | 4 of 12 | 12,288 | 614.4/n |

注释：
1、n是帧的持续时间（表示为20毫秒的整数倍）。对于55比特的帧，n = 1或2，对于长度大于55比特的帧，n = 1，2或4。
2、5毫秒帧只适用于前向基本信道，另外，前向基本信道只能使用n = 1时21至267比特长度的帧。
3、大于552比特的前向补充信道可以采用Turbo编码方式，除此之外只能采用K = 9的卷积编码器。
4、卷积编码需要8比特的填充位，Turbo编码8比特填充位的前两个比特是保留位，其余的6比特是Turbo编码器产生的数据。

图 10-35　RC5 前向基本信道和前向补充信道

与 cdma 2000 移动台相似地，基站发射机也有一个长码产生器，用于对交织后的前向业务信道数据帧实施扰码，如图 10-36 所示。长码产生器的输出信号是一个码片速率为

1.2288Mchip/s 的序列，序列长度等于 $2^{42}-1$。基站发射机按照交织信号的数据率从长码序列中抽取数据得到一个序列，然后把这个序列作为扰码序列与交织信号执行异或操作，从而实现对信道数据的扰码。

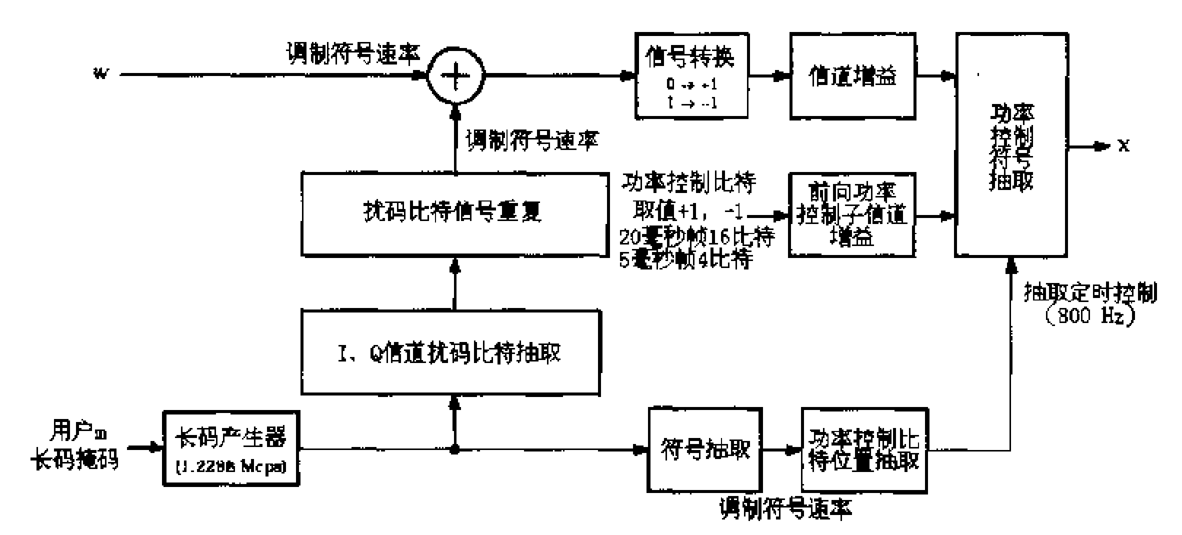



图 10-36　RC3~5 前向业务信道的扰码和功控

另外，对于前向基本信道和前向专用控制信道，cdma 2000 还设置了一个功率控制子信道，用于实现对移动台发射功率的控制。功率控制子信道是一个由 0 和 1 组成的序列，其中 0 表示移动台应该提高发射功率，1 表示移动台降低发射功率。功率控制子信道的最高传输速率可以达到 800bit/s，每个功率控制位将取代前向基本信道或前向专用控制信道的若干个交织信号。功率控制位插入前向业务帧中的位置也有长码信号序列决定，具体说来，基站发射机从长码序列中抽取出得到一个二进制序列，并且根据这个序列中的某些数据位来确定下一个功率控制位的插入点。前向补充信道则不需要插入功率控制子信道。

cdma 2000 前向信道有两种工作模式：OTD（Orthogonal Transmit Diversity）模式和非 OTD 模式（non-OTD）。OTD 模式把前向信道符号分配到不同的发射天线中，并且根据各个天线的 Walsh 码或准正交函数对信道符号进行扩频和调制。非 OTD 模式则没有这个过程，它只是简单地把信道符号轮流分配到两个支路上，如图 10-37 所示。

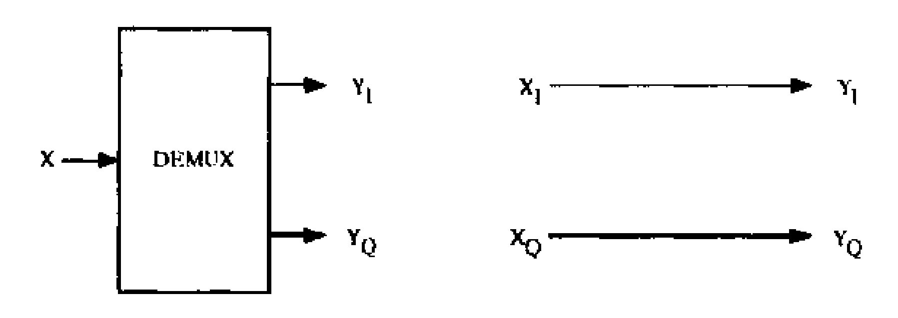



图 10-37　非 OTD 模式前向业务信道的解复用

在非 OTD 模式中，信道符号分成两个支路（$Y_I$和$Y_Q$），这两个支路的符号分别与两个短

PN 序列（$PN_I$ 和 $PN_Q$）相乘，由此得到的两个支路数据再分别通过一个基带滤波器，经过滤波和调制得到射频输出信号，如图 10-38 所示。

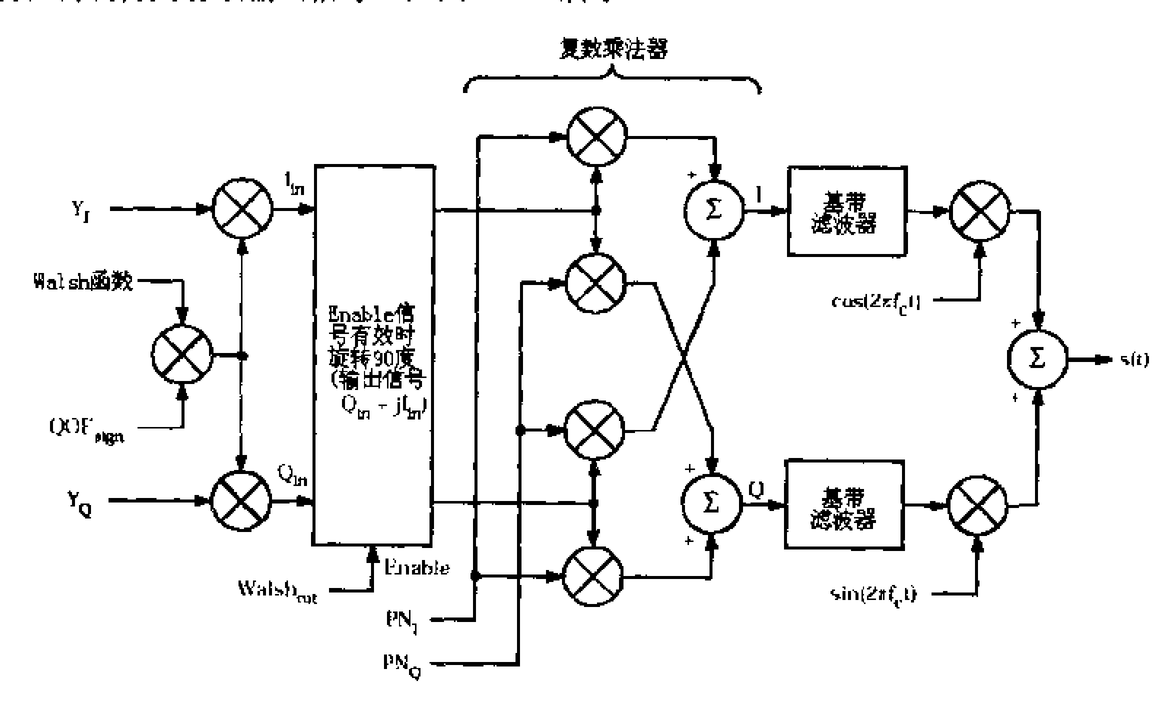



图 10-38　前向业务信道的信号调制

在 cdma 系统中，基站给每个前向业务信道分配一个唯一的 Walsh 码，因此，基站能够支持的业务信道数目受到 Walsh 码个数的限制。为此，cdma 2000 采用了准正交函数（QOF, Quasi-Orthogonal Function），以次来扩展前向业务信道的数量。准正交函数 $QOF$ 是用一个一个准正交函数掩码 $QOF_{sign}$ 去乘以一个正交函数 $Walsh$，再经过 Walsh 旋转函数 $Walsh_{rot}$ 的变换后得到的序列。Walsh 旋转函数 $Walsh_{rot}$ 可以看作是一个具有两个输入端口和两个输出端口的变换模块，函数的取值范围是 1 和 $j$：当 $Walsh_{rot}=1$ 时，两个输入支路的信道符号（$I_{in}$ 和 $Q_{in}$）直接从两个输出端口中输出；当 $Walsh_{rot}=j$ 时，这两个支路的信号被旋转 90°，即第二个支路的信道符号 $Q_{in}$ 变换成 $-Q_{in}$ 之后在第一个端口输出，第一个支路的信道符号 $I_{in}$ 则通过第二个端口输出。